UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Técnicas de geolocalización para apps React y React Native basadas en Push Notification

---

Matias Szarfer Barenblit

Director: Jorge García Vidal

Universitat Politècnica de Catalunya - Facultat d'Informàtica de Barcelona

Tecnologías de la Información

Trabajo de Final de Grado

Otoño 2019

Resumen

El objetivo de éste trabajo se basa en descubrir las posibilidades de geolocalización basadas en notificaciones push para aplicaciones híbridas como método de geolocalización. Se diseña una librería de sencilla integración que permite a un desarrollador captar datos de localización de un dispositivo móvil, y guardar sus registros en una base de datos.

Resum

L'objectiu d'aquest treball es basa en descobrir les possibilitats de geolocalització basades en notificacions push per aplicacions hibrides com a mètode de geolocalització. Es dissenya una llibreria de senzilla integració que permet a un desenvolupador captar dades de localització d'un dispositiu mòbil, i guardar els seus registres a una base de dades.

Abstract

The goal of this project is to discover the possibilities of geolocation based on push notifications for hybrid applications as a geolocation method. An easy-to-integrate library is designed. It allows a developer to capture location data from a mobile device, and store its records in a database.

Índice

1 - Introducción y contextualización

1.1. Introducción

La idea de este Trabajo de Final de Grado surge de un proyecto que me encargó un equipo de investigación de la Universidad Politécnica de Cataluña. Resultó ser un proyecto muy interesante, con la integración de múltiples tecnologías diferentes y un amplio campo de estudio.

Es por estas razones que decidí trasladar el proyecto realizado, a mi TFG, para tener la oportunidad de analizarlo más detalladamente y realizar una investigación más profunda.

A lo largo de este documento se explicará de dónde surge la idea, se contextualizará en su marco, se definirá el proyecto detalladamente y se justificarán las decisiones tomadas. Se expondrá también la planificación inicial, seguida de una evaluación en el momento de su finalización. Revisaremos la metodología y herramientas usadas, así como una detallada explicación de cómo se ha llevado a cabo y gestionado este proyecto. Se incluirá también un análisis económico, social y medioambiental, así como una descripción su funcionamiento y las leyes que afectan a este trabajo.

1.2. Contexto

En esta primera sección del Trabajo de Final de Grado (TFG) se hace una introducción al proyecto desarrollado; se explicará el **contexto** del mismo y el marco del cual **surge la idea**. Se entenderá en qué consiste el plan en el cual se profundizará más en las futuras secciones.

1.2.1. Marco del proyecto

Éste trabajo de final de grado se encuentra en el marco del grupo de investigación **SANS** (*Statistical Analysis of Networks and Systems*) de la **Universidad Politécnica de Cataluña**. Las actividad de investigación del grupo SANS "*se centra en la captura y análisis estadístico de datos provenientes de dispositivos IoT, aplicados a diferentes áreas como la monitorización medioambiental, las ciudades inteligentes y el análisis de patrones de movilidad. Las actividades de investigación actuales se centran en las siguientes áreas: Calibración de redes de sensores, técnicas de aprendizaje de*

máquinas, análisis de tráfico móvil, Internet de los objetos (IOT) y aplicaciones de Smart Cities, y seguridad en redes de sensores." [1]

En este contexto, el grupo SANS está interesado en **procesar datos de geolocalización para de ahí deducir trayectorias de zonas de estancia.** A partir de esta información, por ejemplo, se puede caracterizar los patrones de movilidad de ciertos **colectivos** (por ejemplo, mujeres embarazadas, ancianos, enfermos crónicos…) para cruzarlos con mapas de contaminación. Según indica el profesor Jorge García Vidal, **responsable** del grupo de investigación y **director** de éste TFG, uno de los puntos más difíciles en este proceso es la obtención de datos fiables de geolocalización.

El proyecto explora un método de obtención de información de geolocalización basado en **Notificaciones Push** (*Push Notifications*) para dispositivos móviles. En el caso de que los resultados obtenidos sean positivos, la intención del grupo de investigación SANS es incorporar esta técnica **en varios proyectos de investigación**.

1.2.2. Definición del trabajo

El proyecto se titula "*Técnicas de geolocalización para apps React y React Native basadas en Push Notification*". El proyecto investiga el problema ya comentado anteriormente; la **captación de datos de geolocalización**. El producto final de este proyecto es una librería de geolocalización que permite al desarrollador activar funciones de geolocalización en una aplicación en React[1] o React Native[2] que son pedidas por un servidor mediante la técnica de Push Notification[3]. La librería dispone de unas llamadas que permiten **activar, desactivar o modificar el servicio**.

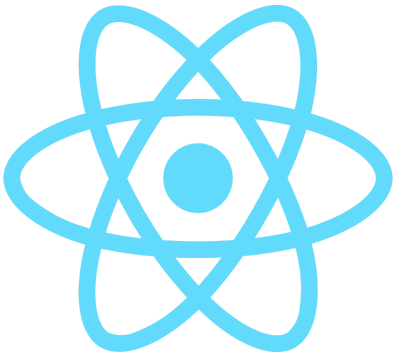

Figura 1. Logo de React y React Native (fuente: reactjs.org)

[1] Framework JavaScript para crear aplicaciones Web basado en componentes reutilizables [11].

[2] Framework JavaScript para crear aplicaciones nativas para iOS y Android basadas en React [12].

[3] Al contrario de las Pull Notifications, en la que el cliente pide al servidor actualizaciones mediante polling, en esta técnica es el servidor quien informa al cliente de las nuevas notificaciones [10].

1.2.3. Problema a resolver

Para la recolección de los datos para su posterior tratado, se decidió crear una aplicación **híbrida**[4] compatible con **iOS, Android y Web**. El proyecto no implementa la aplicación completa, sino que ofrece **una librería** para la fácil integración de esta compleja tarea mediante llamadas a una API.

El problema que el equipo SANS necesitaba resolver es sencillo de explicar: Necesitaban implementar una librería para que, **desde un servidor** y mediante **Push Notifications**, se **guarden** los datos de geolocalización de un dispositivo móvil, mientras la aplicación se ejecuta en **segundo plano**.

1.2.4. Actores implicados

La librería va destinada a un **público técnico**, pero que no específicamente serán desarrolladores expertos de React ni React Native, pero sí del ámbito de aplicaciones híbridas y programación web.

Debido a que es un sistema de recolección de datos, naturalmente necesitaremos individuos que, **voluntariamente** instalen la aplicación para ayudar a la recolección de posiciones. Los registros guardados de estas personas serán posteriormente **analizados por el equipo de investigación.**

Finalmente, los que se beneficiarán de sus resultados será el equipo de investigación para su posterior tratado, y a largo plazo, la **sociedad en general**.

1.2.5. Arquitectura

Explicaremos a continuación en qué consiste la arquitectura de la aplicación. Para hacerlo más claro, se incluye la siguiente imagen (figura 2) que muestra de forma muy simplificada la arquitectura a seguir.

Como *endpoints*, tenemos un teléfono *smartphone* que puede ser **tanto iOS como Android**, o ejecutar el programa que implementa la librería en una **página web**. La forma de recibir una petición de geolocalización, será mediante notificaciones push, que el servidor de **Firebase** enviará al dispositivo (leer justificación a continuación). Los teléfonos contestarán a esta notificación enviando un paquete de datos, que firebase guardará en una **base de datos en tiempo real**. Finalmente, quien dará las órdenes de envío de notificaciones push será el técnico mediante un servidor propio que se ejecutará en un entorno privado.

[4] Una aplicación híbrida es aquella que combina tecnología web y nativa para su desarrollo [9].

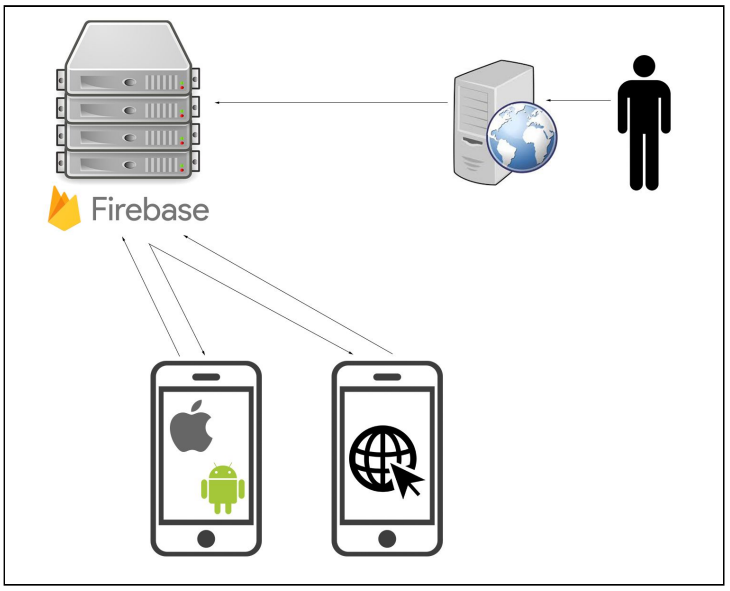

Figura 2. Esquema de la arquitectura planeada

1.3. Justificación

Se han intentado buscar soluciones ya existentes, pero **no se ha encontrado ninguna** que se adapte a lo que nosotros queremos. A continuación, se citan las diferentes fuentes consultadas, que son las páginas web más populares en cuanto a componentes de React y React Native se trata:

- awesome-react-native.com/#geolocation
- js.coach/?search=geolocation
- Búsqueda en Google: "React Geolocation Push Notifications"

En ninguna de las tres fuentes se obtienen resultados relacionados con Push Notifications. Sin embargo, sí hay resultados para **Background Jobs y demás técnicas de polling**. Por lo tanto, se decide realizar una **librería propia**.

Cómo hemos especificado, necesitamos también un servidor que envíe Push Notifications. Básicamente teníamos tres opciones: podíamos crear **nuestro propio servidor** y sistema de Push Notifications, usar Amazon Web Services [2], o utilizar Google Firebase [3]. El primero se descarta ya que tiene una complejidad excesivamente grande. Finalmente decidí **usar Google Firebase** debido a que tengo tengo más conocimientos del entorno, habiendo trabajado con Firebase en el pasado. Se usará tanto como servidor de Push Notifications como de **base de datos** para almacenar los registros de posición.

Finalmente, para implementar las Push Notification, se tomó información de proyectos ya existentes [4] [5], y se llegó a la conclusión que la mejor alternativa, ya que nos interesa que la aplicación **funcione en segundo plano**, es una librería multiplataforma llamada **React Native Firebase** [6]. La misma sirve para Android y

iOS. Sin embargo, para la versión web se opta por utilizar la **librería Firebase para JavaScript proporcionada por Google** [3] debido a su extensa documentación, compatibilidad con React y múltiples tutoriales.

2 - Planificación

2.1. Alcance

Antes de empezar el proyecto, se realizó una **reunión con el director** para debatir y decidir el alcance. Se respondió a las preguntas de qué cosas queríamos abarcar, y qué cosas **no eran necesarias**. Para ello, se definieron unos objetivos, tanto técnicos como personales y académicos. Se definieron unos requerimientos, que serían las funciones que más adelante querríamos tener implementadas. Finalmente se hizo también una **evaluación de riesgos y obstáculos** que podríamos encontrar.

2.1.1. Objetivos y sub-objetivos

A continuación, se presentan los objetivos y sub-objetivos del proyecto tanto a profesional, académico y personal. Se incluye también el **desglose** de estos.

1. **Resolver el problema**
    a. Crear una librería
        i. Hacer que se ejecute en segundo plano.
        ii. Activar y desactivar la recepción de notificaciones.
        iii. Ocultar las notificaciones visibles.
        iv. Enviar la información de posición a un servidor.
        v. Hacer que la librería sea modular y *customizable.*
        vi. Crear una documentación completa de la librería.
        vii. Hacerla integrable con Web, Android y iOS.
        viii. Hacer *testing* y experimentos.
        ix. Permite añadir y leer un nombre de usuario y un *token*.
    b. Crear una aplicación modelo
        i. Conseguir que se ejecute en segundo plano.
        ii. Crear un ejemplo sencillo.
        iii. Crear una documentación completa.
    c. Configurar un servidor FCM
        i. Programar el envío de notificaciones a un dispositivo periódicamente.

   ii. Aprender como funciona Node.JS.
2. **Conocimientos a nivel personal**
   a. Planificación
      i. Aprender a realizar una planificación de un proyecto.
      ii. Describir la naturaleza, alcance y objetivo del proyecto.
      iii. Aprender a valorar los tiempos, tareas y crear un Gantt.
      iv. Identificar y reducir los riesgos.
   b. Presentar una tesis completa
      i. Aprender a escribir de forma correcta un documento.
      ii. Gestionar las referencias.
      iii. Crear un informe ambiental, social y económico del proyecto.

2.1.2. Requerimientos

Se requieren las siguientes funciones. Se incluye también qué objetivos cumplen.

- **StartService:** Activa la recepción de Push Notifications en segundo plano, y cuando llega una, envía la posición actual a una base de datos (1.a, 1.b, 1.c, 1.d, 1.g)
- **StopService:** Detiene la recepción de Push Notifications (1.b).
- **GetUsername:** Devuelve el nombre de usuario (1.i, 1.e).
- **GetToken:** Devuelve un *token* (1.i, 1.e).
- **SetUsername:** Establece un nombre de usuario y lo guarda en la memoria (1.i, 1.e).
- **SetToken:** Establece un *token* y lo guarda en la memoria (1.i, 1.e).
- **Aplicación modelo:** Hecha tanto para React como React Native, consta de un botón de Empezar y Detener. Incluye funciones de muestra para modificar y leer *tokens* y usuarios. Debe ejecutarse en segundo plano (2.a, 2.b).
- **Servidor de FCM:** Debe enviar periódicamente notificaciones a un dispositivo mediante una interfaz (3.a, 3.b).
- **TFG:** Crear, a posteriori, una memoria para presentar como trabajo de final de grado (4).

2.1.3. Obstáculos y riesgos

Por supuesto, nos encontramos con varios obstáculos y riesgos que son importante evaluar y estimar desde el principio. Lo más complicado, según se pudo ver por **falta de documentación**, sería no conseguir que una aplicación híbrida funcione en **segundo plano**. Otros riesgos pueden ser no conseguir que funcione la propuesta. También que no se pueda hacer, por motivos técnicos, para iOS. Cómo se puede ver en la documentación [7], el segundo plano **no funcionará** para iOS. Sin embargo, hacerlo funcionar para el sistema operativo del iPhone no era una prioridad.

Obstáculos también son que la programación para iOS tiene que estar hecha con *hardware* de Apple, y por desgracia **no se dispone** del mismo. El sistema de la FIB no permite tampoco la instalación de aplicaciones y librerías, por lo que, por un tema de recursos, puede ser complicado conseguir una aplicación funcional para iOS.

Entre los riesgos se valora también el principio de la WWW que no permite (en Junio de 2019), que una aplicación web **recopile información de geolocalización en segundo plano**. Se recomienda leer la discusión del W3C Consortium [8].

3 - Metodología y rigor

3.1. Metodología

Se ha decidido seguir una metodología por etapas y reuniones semanales con el responsable de la investigación, que es la misma persona que el director del TFG (Prof. Jorge García). Se decide que cada miércoles se realizará una **reunión de seguimiento** para ver como va la implementación de la librería, así como para decidir nuevas especificaciones en caso de que sea necesario. También se decide **separarlo en dos fases** de desarrollo y una para la memoria del TFG, y dentro de cada fase en diferentes etapas. La idea es que al final de cada etapa, se pueda ver y probar algo funcionando.

Primera fase: MPV[5]

- Aplicación básica Android (5 horas)
- Recepción de notificación FCM (8 horas)
- Guardar la localización en un log (7 horas)
- Correr la aplicación en segundo plano (7 horas)

Segunda fase: Detalles

- Conexión con Firebase Database (10 horas)
- Envío de localización a la BD (10 horas)
- Añadir parámetros (15 horas)
- Adaptarlo a iOS (20 horas)
- Adaptarlo a web (15 horas)
- Programar un servidor para el envío de mensajes (15 horas)

Tercera fase: TFG

- Realización de experimentos (1 semana)
- Escritura de la memoria

Como se puede ver, se desarrollará la librería **primero para Android**, ya que es lo más importante, y **luego se adaptará** a iOS y web. Tampoco se programará el servidor desde el principio, sino que las notificaciones se harán **temporalmente** a mano mediante peticiones POST a la URL de Firebase, hasta tener el servidor.

[5] Por sus siglas en inglés, el Producto Mínimo Viable es la primera fase de un producto que permite su utilización.

3.1.1. Herramientas usadas

Para la correcta gestión del proyecto, la comunicación con el responsable, y una mejor coordinación de las diferentes tareas, se especifica a continuación las diferentes herramientas que se usarán durante el proyecto.

Gestión

Correo: Para la comunicación diaria con el profesor Jorge García Vidal, se usarán correos electrónicos.

Google Calendar: Utilizamos el calendario de Google para concretar reuniones semanales, modificarlas o cancelarlas en caso de necesidad.

Hangouts: Una de las herramientas más importantes de la gestión. Debido a que yo me encontraré en un intercambio universitario durante el desarrollo del proyecto, las reuniones semanales se harán remotamente. Utilizaremos las videollamadas para comunicarnos una vez por semana.

Tareas

Trello: La aplicación ofrece una interfaz con tarjetas que son fáciles de mover de un sitio a otro. Se utilizarán diferentes paneles para llevar un control de las tareas pendientes, en proceso y hechas. Es **indispensable** indicar el tiempo de cada tarea para la mejor organización del desarrollador. En la imagen se ve el estado de la tabla en un momento del desarrollo.

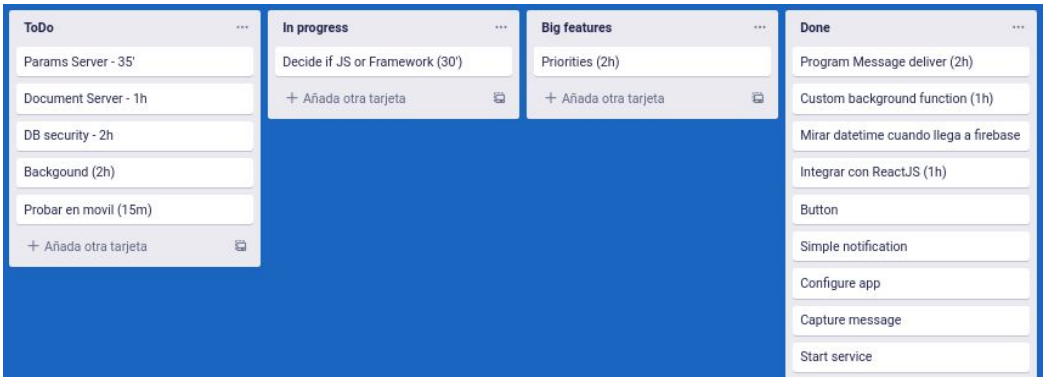

Figura 7. Captura de pantalla de Trello

Gestión de versiones

GitHub: El almacenaje del código y la gestión de versiónes se realizarán mediante Git, usando GitHub como proveedor. Existirá una rama (*branch*) **master**, y se irán creando ramas para cada nueva función. Después de probar que funciona

correctamente, se hará *merge* en la rama master. De esta forma evitamos que el código principal pueda quedar inutilizable.

3.2. Descripción de las tareas

Se describe a continuación las tareas necesarias para alcanzar los objetivos propuestos anteriormente. Se ha decidido buscar una granularidad muy fina, para un seguimiento del proyecto más eficiente. Como se ha comentado anteriormente, con la herramienta Trello será sencillo verificar el progreso del proyecto usando las tareas que se describen en esta sección.

3.2.1. Detalle de tareas

Para realizar la descripción de tareas nos remontamos a la lista detallada de la primera entrega. A continuación se explicarán las sub-tareas.

Primera fase: MPV

Aplicación básica Android (5 horas)

- **T1 – Setup del proyecto (1h):** Crear repositorio en GitHub, clonar el modelo de la librería, instalar dependencias e instalarlo en el dispositivo.
- **T1.1 – Setup del dispositivo (30'):** Habilitar el modo desarrollador, instalar drivers necesarios en el PC, habilitar depuración.
- **T2 – Pantalla principal (1h):** Crear un botón, añadir un texto informativo y habilitar los logs mediante la consola.
- **T2.1 – Creación de clases (2h):** Creación de los diferentes archivos y clases, escribir las funciones para ser llamadas, y crear así una pre visualización del código.
- **T3 – Reunión de seguimiento (30')**

Recepción de notificación FCM (8 horas)

- **T4 – Crear notificación de prueba (2h):** Investigar cómo enviar una notificación a un dispositivo, conseguir el *token* del Smartphone, configurar Postman [13].
- **T4.1 – Recepción (2h):** Activar la recepción de notificaciones mediante JavaScript.
- **T4.2 – Tratado (2h):** Tratar la notificación al llegar y decidir qué hacer, dependiendo de los campos que la misma tenga.
- **T4.3 – Detención (1h):** Crear la función que detiene la llegada de notificaciones.

- **T5 – Conexión con vista (30'):** Conectar las funciones con las vistas hechas anteriormente.
- **T6 – Reunión de seguimiento (30')**

Guardar la localización en un log (7 horas)

- **T7 – Investigación (2h):** Investigar cómo obtener la geolocalización nativa desde el código JavaScript.
- **T8 – Obtención de geolocalización (3h):** Crear una función que obtiene la geolocalización del dispositivo.
- **T8.1 – Log (1h):** *Loggear* la información de geolocalización en la consola y analizar su exactitud.
- **T9 – Reunión de seguimiento (1h)**
- Ejecutar la aplicación en segundo plano (7 horas)
- **T10 – Investigación (2h):** Investigar cómo ejecutar una aplicación en React Native en segundo plano y permitir la recepción de notificaciones.
- **T11 – Worker (2h):** Crear un *worker*[6] que haga el trabajo de la T4 en segundo plano.
- **T11.1 – Geolocalización (1h):** Adaptar el *worker* para que haga la T8 en segundo plano.
- **T12 – Testing (1h30'):** Hacer un testeo de las funciones implementadas en segundo plano.
- **T13 – Reunión de seguimiento (30')**

Segunda fase: Detalles

Conexión con Firebase Database (6 horas)

- **T14 – Creación de BD (3h):** Crear una base de datos en la nube, elegir modelo, investigación sobre las ventajas.
- **T15 – Integración (3h):** Integrar las herramientas de Google Firebase en la aplicación, conseguir el *token*, configurar los parámetros.
- Envío de localización a la BD (9 horas)
- **T16 – Investigación (2h):** Investigar la mejor forma de controlar la BD remotamente desde el cliente.
- **T17 – Funciones (4h):** Crear las funciones de envío de información a la base de datos y modificar las existentes para que el log se haga en la BD.
- **T18 – Testing (2h):** Comprobar que el funcionamiento es el esperado.
- **T19 – Reunión de seguimiento (1h)**

[6] "Los Web Workers dedicados proveen un medio sencillo para que el contenido web ejecute scripts en hilos en segundo plano. Una vez creado, un worker puede enviar mensajes a la tarea creada mediante envio de mensajes al manejador de eventos especificado por el creador." [14]

Añadir parámetros (15 horas)

- **T20 – Planificación (2h):** Decidir qué parámetros se querrán añadir y permitir modificar al usuario (desarrollador).
- **T21 – Storage (4h):** Configurar una clase de almacenamiento para guardar los parámetros en el dispositivo.
- **T22 – Creación de funciones (4h):** Crear las funciones para modificar y leer los parámetros.
- **T23 – Añadido de información a la BD (4h):** Añadir estos parámetros a las funciones que envían información a la BD.
- **T24 – Reunión de seguimiento (1h)**

Adaptarlo a iOS (19.5 horas)

- **T25 – Conseguir materiales (2h):** Conseguir equipos de entorno Apple.
- **T26 – Configuración de equipos (4h):** Configurar e instalar software necesario.
- **T27 – Investigación (2h):** Hacer una búsqueda sobre los parámetros a cambiar entre Android e iOS.
- **T28 – Adaptación (11h):** Adaptar las funciones ya explicadas a los cambios encontrados en la T27.
- **T29 – Reunión de seguimiento (30')**

Adaptarlo a web (15 horas)

- **T30 – Configuración de entorno (3h30):** Configurar el entorno ReactJS, instalar parámetros, aprovechar código (vistas, funciones, etc)
- **T31 – Investigación (1h):** Investigar qué aspectos habrá que adaptar.
- **T32 – Adaptación (10h):** Modificar las funciones pertinentes, eliminar *worker*, etc.
- **T33 – Reunión de seguimiento (30')**

Programar un servidor para el envío de mensajes (15 horas)

- **T34 – Configuración de entorno (2h):** Configurar el entorno Node.JS, instalar dependencias y software necesario.
- **T35 – Creación de función (2h):** Creación de una función de envío de mensajes mediante *Push Notification*.
- **T35.1 – Programación de envío (4h):** Creación de una función que llame de forma recurrente a la función de la T35.
- **T36 – Parametrización (4h):** Permitir al usuario especificar los parámetros (frecuencia, mensaje, destinatario…) ya sea mediante un fichero de configuración o parámetros individuales.
- **T37 – CLI (2h30):** Crear un CLI (*Command Line Interface*) agradable y fácil de utilizar.

- **T38 – Reunión de seguimiento (30')**

Tercera fase: TFG

Realización de experimentos (1 semana)

- **T39 – Planificación (2h):** Decidir cómo se llevarán a cabo los experimentos.
- **T40 – Ejecución (168h):** Ejecutar los experimentos durante 1 semana.
- **T41 – Análisis de datos (5h):** Analizar los datos obtenidos y crear informe.

Escritura de la memoria

Ver el anexo 10.2 para una tabla resumida.

3.3. Estimaciones y Gantt

Se adjunta también el diagrama de Gantt, que nos permite entender en profundidad cómo se desarrollará el proyecto a lo largo del tiempo. *Ver el anexo 10.1 para el gráfico de Gantt.*

3.4. Gestión de riesgos

En todo proyecto siempre surgirán riesgos. Por eso, en esta sección se ha hecho un **análisis de los posibles contratiempos** que podrían surgir. Se han debatido con el equipo para proponer soluciones alternativas y así cumplir los objetivos en el mínimo tiempo posible, y acabar el proyecto en el tiempo planificado.

3.4.1 Tareas alternativas

Se adjunta a continuación una tabla de evaluación de riesgos.

Tarea original	Riesgo	Tarea alternativa	Tiempo extra (h)	Recursos extra
T11 – Worker	No conseguir la ejecución en segundo plano en App híbrida mediante Push Notifications	T11a – Wakeup timer: Crear un temporizador que ejecute un job regularmente	5h	-
T25-29 – iOS	Podría ser que no se consiga adaptar la aplicación a iOS	En este caso se evaluará la necesidad de habilitarlo para iOS		-

T32	WWW no permite geolocalización en segundo plano	Se dejará para que se ejecute sólo en primer plano	0h	-

Tabla 2. Evaluación de riesgos

3.5. Presupuesto

Todo proyecto tiene un coste, y éste no es una excepción. Si bien no hay demasiados costes directos más allá de la **mano de obra**, es importante remarcar cuáles son estos. También se detalla los costes **indirectos** de hardware y demás recursos.

3.5.1. Identificación y estimación de costes

3.5.1.1. Recursos humanos

Se adjunta a continuación la planificación de costes de recursos humanos. Dado que el autor del TFG es realiza a la vez los roles de programador, analista y diseñador, **no se ha separado por roles**.

Stage	Tiempo (h)	Precio/h	Total (€)
Aplicación básica android	5	10	50
Recepción de notificaciones FCM	8	10	80
Registro de localización en log	7	10	70
Ejecución en segundo plano	7	10	70
Conexión Firebase	6	10	60
Envío a la base de datos	9	10	90
Adición de parámetros	15	10	150
Adaptación a iOS	19.5	10	195
Adaptación a web	15	10	150
Programación de servidor	15	10	150
Total	**106,5**	**10**	**1.065 €**

Tabla 3. Fraccionado de precio por etapas del proyecto

3.5.1.2. Recursos materiales

Se añade también una estimación del coste de desarrollo en material hardware para conseguir una mejor aproximación del gasto económico total.

- Acer F5 i7 128+500 GB ROM / 8 GB RAM - Ordenador Personal Linux
- OnePlus 5T 64 GB ROM / 4 GB RAM - Móvil personal Android
- iPhone 7 128 GB ROM / 2 GB RAM - Móvil prestado iOS
- iMAC 2012 i5 1TB ROM / 8 GB RAM - Equipo Mac Prestado

Producto	Coste (€)	Vida útil (años)	Tiempo usado (h)	Amortización (€)
Acer F5	799	5	106,5	1,95
OnePlus 5T	489	4	200	2,79
iPhone 7	769	4	200	4,38
iMac 2012	1246	6	20	0,47
Total				**9,59**

Tabla 4. Coste de amortización del hardware propio

3.5.1.3 Recursos software

Finalmente se cita el software utilizado y sus respectivas licencias.

- Github - Gestión de versiones - Licencia estudiante.
- IntelliJ WebStorm - IDE de programación - Licencia estudiante.
- Visual Studio Code - IDE de programación - Licencia gratuita.
- SublimeText - Editor de código - Licencia gratuita.
- WPS Community - Editor de textos - Licencia abierta.
- LaTEX - Editor de textos - Licencia abierta.

Elemento	Precio (€)
Github	0
WebStorm	0
Visual Studio Code	0
SublimeText	0
WPS Community	0
LaTEX	0
Total	**0 €**

Tabla 5. Precio del software utilizado

3.5.1.4. Contingencias e imprevistos

Cómo se ha comentado anteriormente, siempre puede haber cosas que salgan mal o **no vayan según lo previsto**. Por ello, es necesario tenerlo en cuenta y preverlo a la hora de realizar un presupuesto económico.

Para los imprevistos, nos remontamos al **punto 3.2**, donde se calcula que el total de horas extra que se pueden llegar a necesitar por razones de contratiempos, son 5 horas. Por ello calculamos añadir 50€ de imprevistos (5h a 10€/h). También se debe tener en cuenta el **extra de uso de hardware**, aunque es prácticamente **insignificante** (0,09€).

Finalmente se dejará también un 10% del **presupuesto total para contingencias**, y cosas que no hayamos previsto que podían salir mal.

3.5.1.5. Coste total

La planificación económica final es, pues, la siguiente tabla:

Elemento	Coste (€)
Recursos humanos	1.065
Hardware	9,59
Software	0
Imprevistos	50,09
Contingencias	112,46
Total	**1237,14 €**

Tabla 6. Coste total del proyecto

3.5.2. Control de gestión

Lo que más puede variar en este trabajo que necesita pocos recursos físicos, es el factor humano. Mediante las **reuniones semanales** se irá rectificando las tareas que hayan tardado más, buscando la mejor solución para no alargar el proceso.

3.6. Cambios en la metodología

En general, se ha seguido la metodología establecida. **No se ha visto la necesidad de cambiar ninguno de los aspectos,** ya que nos hemos dado cuenta que nuestro método iterativo de reuniones semanales, **funcionaba perfectamente**. Semana a semana, íbamos iterando. Modificamos los aspectos que **no habían funcionado** de la semana anterior, y definimos **nuevos objetivos** para la próxima reunión.

Este método, además de ser eficiente en el aspecto del desarrollo, fue útil a nivel humano: Al ser un solo desarrollador, puede ser difícil a veces hacer un seguimiento y ponerse unos objetivos. Es por eso, que **con la ayuda de otra persona**, y mediante la discusión y el debate, surgen ideas mucho **más ricas**.

En cuanto a los costes, tal y como se comenta en el **punto 3.7.2.**, no han habido costes extra. Para las únicas modificaciones que han habido se ha decidido sacarlas del plan, por lo que el coste final ha sido el estimado, sin las contingencias ni imprevistos: **1.065,00 euros**.

3.7. Desviaciones finales

Una vez finalizado el proyecto, y mirándolo con perspectiva, se hace un análisis retrospectivo de los aspectos que se han desviado de la planificación inicial.

3.7.1. Resultado final

El resultado final es exitoso y corresponde en su mayoría con la planificación inicial. A la hora de entregar el proyecto, dispone de los distintos elementos requeridos por el grupo de investigación.

Aplicación: Se creó una aplicación en React Native que funciona **sólo para Android**, y muestra el ejemplo de la librería. Con esta aplicación, se puede encender y apagar el servicio, que irá recogiendo datos de localización con una cierta frecuencia. La aplicación se ejecuta en segundo plano sin dificultades y durante largos períodos de tiempo **sin detenerse**. Envía **paquetes de datos** a una base de datos como se había especificado. En la imagen se ve una captura de pantalla con el servicio apagado y encendido.

Figura 3. Captura de pantalla de la aplicación ejemplo (servicio apagado y encendido)

Servidor: El servidor, equipado con una interfaz mediante comandos (CLI), es sencilla y fácil de usar. Envía peticiones de Push Notification al servidor de Firebase. Éste **envía las Push Notification al token especificado**. La aplicación de Servidor es totalmente **parametrizable**. Se pueden especificar las variables en un **fichero de configuración**, o mediante el **CLI** (*Command Line Interface*). En la imagen se observa el envío exitoso de 14 notificaciones push a un dispositivo, con un intervalo de 10 segundos. El *token*[7] y la clave del servidor están especificadas en el **fichero de configuración**.

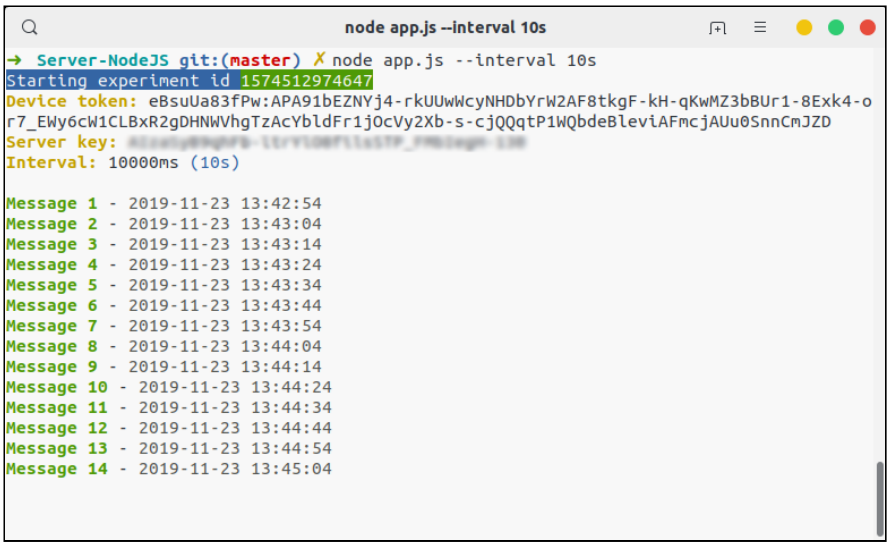

Figura 4. Captura de pantalla de el CLI del servidor[8]

[7] Recordemos que el token es una cadena de caracteres que permite al servidor de Firebase identificar un dispositivo para enviar una notificación push.

[8] La clave del servidor ha sido difuminada ya que con ella se pueden acceder a todos datos almacenados en la Base de Datos.

Aplicación web: La librería está adaptada para funcionar en un navegador en **primer plano**. Con las mismas especificaciones que la aplicación nativa, recolecta de forma satisfactoria la localización del dispositivo. En la imagen se muestra una captura de pantalla de la página web con el servicio apagado.

Figura 5. Captura de pantalla de la aplicación versión web

Base de datos: Usando Firebase, se almacenan de forma indexada los datos de cada sesión de usuario. Eso significa que cada vez que el usuario activa la librería, se crea una **nueva rama** en la base de datos. Dentro de la misma, se guardan todos los registros **hasta que se detenga el servicio**.

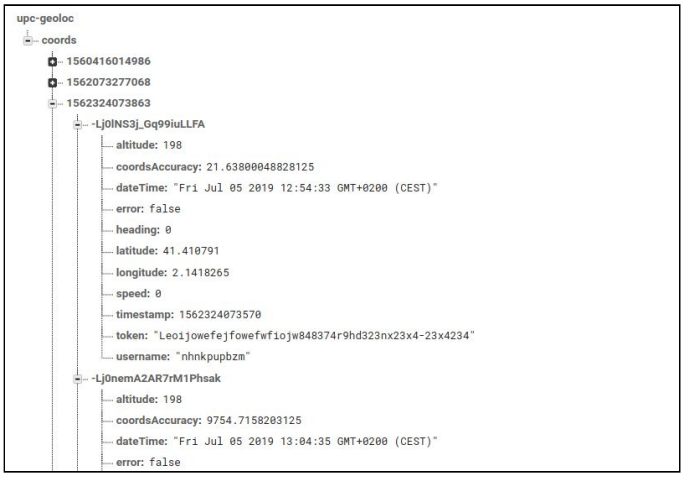

Figura 6. Captura de pantalla de una muestra de datos en la base de datos

3.7.2. Desviaciones

En el momento de la escritura de este documento, el proyecto se encuentra en fase de **finalización y experimentación**. Lo único que faltará para darlo por finalizado con todos sus objetivos, es la escritura del TFG y la recolección y análisis de los datos recolectados en verano para hacer las pruebas, que aún están almacenadas en la base de datos.

Sin embargo, durante la realización del proyecto **aparecieron algunas desviaciones**. Por unas causas u otras, especificadas a continuación, no se pudieron realizar algunos de los requisitos iniciales. Asimismo, se detectaron nuevas necesidades a lo largo del desarrollo, por lo que hubo que añadir **nuevas especificaciones** al proyecto.

iOS

El principal cambio respecto al original es el objetivo **a.vii:** Crear una aplicación compatible con iOS, Android y Web. La adaptación a iOS resultó ser muy costosa. Requería unos costes económicos muy elevados por alquiler de material de la marca Apple, debido a que no es suficiente con un dispositivo iPhone, sino que hace falta también compilar la app en un ordenador Mac. En la **Tabla 1** se especifican los costes extra. Éstos no se habían tenido en cuenta en el cálculo del presupuesto original. En ese momento, después de una reunión con el responsable, se **decidió que la versión para iOS era prescindible**. Entonces se tomó la decisión de no hacer la versión para la marca de Apple. De todas formas, esto **se había tenido en cuenta en la planificación**, por lo que no fue una decisión difícil. Ver el **punto 3.4.1.**

Producto	Coste diario	Días en uso	Coste total
iMac [15]	38,40€	7[9]	269€
iPhone [16]	25,40€	7[10]	178€
Total			**447**

Tabla 1. Costes de alquiler de material Apple

Web

Otro aspecto que cambió respecto a la versión original fue la web. Al principio se deseaba que la web se **ejecute en segundo plano**. Como se comenta en el punto 2.1.3, no resultó posible debido a que en la web, no es posible que un *worker* pida la localización en segundo plano. Este aspecto se había tenido en cuenta en la

[9] 19,5h planificadas para esta tarea. 5h/día = 4 días. Añadimos dos de testeo y uno de margen.
[10] 19,5h planificadas para esta tarea. 5h/día = 4 días. Añadimos dos de testeo y uno de margen.

planificación inicial (punto 3.4.1), y por lo tanto supimos **reaccionar a tiempo**, no gastando más recursos de los necesarios, y **decidiendo que la versión web no** se ejecutará en segundo plano.

Servidor

En un momento inicial, al hacer las estimaciones temporales, no se acabó de entender que hacía falta un **tercer agente** para la coordinación de Firebase. Es decir, el servicio de notificaciones push de Firebase no permite la opción de planificar el envío de notificaciones. Por ello, hizo falta el desarrollo de una aplicación que hiciera la función de servidor planificador, y envíe **periódicamente peticiones** al servidor de **Firebase**.

4. Análisis de alternativas

Al haber acabado ya la programación y testeo del proyecto, se puede ver con una mejor perspectiva las decisiones tomadas. Ya se habló al principio (punto 1.3) de las alternativas en cuanto a librerías y demás, pero aprovecharemos este punto **para profundizar** en las decisiones tomadas a lo largo del proyecto, junto con su valoración final.

4.1. Estado del arte

Hoy en día, numerosas aplicaciones en el mercado necesitan acceder a la ubicación del dispositivo para su correcto funcionamiento. Desde Uber [40] para saber dónde recoger al usuario, hasta Google Maps, que almacena constantemente datos de nuestra ubicación para formar lo que ellos llaman el *Timeline* [41].

Muchas de ellas utilizan los métodos tradicionales de obtención de localización [42], es decir, en el momento que la necesitan, la piden a través de la API[11] de Android.

Sin embargo, el reto al que nos enfrentamos es más complejo que los ejemplos mencionados, ya que no disponemos directamente de la API de Android. En su lugar, sólo podemos acceder a la API de React Native [44], que si bien es muy amplia, tiene también algunas restricciones.

Recordemos que esto lo queremos hacer para que esté disponible en multiplataforma.

[11] Una API ofrece una interfaz al programador para acceder a funciones del sistema operativo [43].

Es por ello que hay menos documentación para el ámbito de nuestro proyecto, especialmente combinado con las notificaciones push, ya que, recordemos, queremos que sea el servidor el que decida cuando obtenemos la ubicación.

En esta sección revisaremos las posibles alternativas y mejores soluciones para este problema.

4.2. Framework

Si bien decidimos escribir el proyecto en **React y React Native**, existen diferentes *frameworks* con el mismo propósito. En un primer momento, valoramos la opción de programarlo con otra plataforma, pero finalmente se tomó la decisión que se ha comentado. A continuación las razones.

Ionic: Ionic [18] era la opción más posible. Debatimos si usar este framework, por su rápido desarrollo. Ionic es una herramienta para desarrollar herramientas multiplataforma basadas en Angular [19], su homólogo web. Ofrece una amplia librería de funciones nativas. Sin embargo, se ejecuta sobre **Cordova** [20], su núcleo y punto de conexión con el sistema nativo. Se detectaron ciertos inconvenientes con esta herramienta[12]: Por un lado, es un sistema **bastante cerrado** y con el modelo *freemium*, por lo que para varios módulos **hay que pagar** y ser un miembro premium. Por otro lado, **no es reactivo**, lo que significa, a grandes rasgos, que la **performance es peor**. En el pasado también Cordova dio muchos problemas. Especialmente para una librería que **requiere de mucho contacto con la API nativa** del teléfono, **Ionic no es lo ideal**.

Nativo: Siempre es una buena opción desarrollar algo directamente en nativo. Sin embargo, dado que queríamos, en un punto inicial, crear una librería híbrida para todas las plataformas, **por definición no tiene sentido** hacerlo directamente en Nativo. Aun así, hay otros aspectos que no me atrajeron desde un principio, como la necesidad de aprender el doble de lenguajes de programación, **renovar mis antiguos** conocimientos de Android SDK, así como lo **lento y tedioso** que es el desarrollo nativo. Al final, lo que buscan todos los *frameworks* híbridos es **reducir el tiempo de desarrollo**.

VueJS: *"Vue (pronounced /vjuː/, like view) is a progressive framework for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable."* [21]. Podría haber sido una buena alternativa, pero ya conociendo React y React Native por trabajos previos, aprender un nuevo lenguaje y todas sus herramientas **resultaba redundante.** Además, ahora mismo

[12] Basado en mi experiencia como desarrollador de Ionic durante meses.

React es el *framework* **más popular** [22]. Por ello no acaba de tener sentido aprender un lenguaje que se usa menos.

Por todas estas razones, se **decidió usar React y React Native**. Yo tenía ya experiencia en React, por lo que la versión nativa no sería difícil de aprender. Además, me interesaba ganar experiencia con éste lenguaje de programación ya que, como se ha comentado, se ha posicionado primero en los *frameworks* más populares del mundo.

4.3. Técnica

Se decidió usar la técnica de las Push Notification ya que era lo que el equipo de investigación requería. Esta técnica tiene la ventaja de que es **el servidor quien pide** los datos al dispositivo cuando se desee. **Permite que sea el controlador** del servidor quien **decida la frecuencia y el momento** en el que quiere capturar los datos.

La alternativa posible era crear una rutina que se repita en segundo plano. Está **activaría un temporizador**, usando la función **`SetTimeout()`** de JavaScript, que cada X milisegundos activa una función. El problema de esta solución es que, como se comentaba, **no nos permite tener el control**. Es posible que en cierto momento queramos obtener más datos por hora, o simplemente dejar de recibirlos. Debido a que la función se encuentra en el dispositivo, nosotros **no somos capaces de modificar la frecuencia**. Tampoco podemos modificar, por ejemplo los datos que se piden. Al contrario, con la técnica de Push Notifications **podemos pedir unos datos concretos** (latitud, longitud, altitud, velocidad…). Si bien en este momento no es necesario, ya que se envían todos los datos que ofrece la API, en un futuro podría ser útil para **reducir el overhead**.

4.4. Librerías

Geolocation, React Native [23]: Librería propia de React Native. Ofrece una interfaz simple. Mediante la llamada `navigator.geolocation.getCurrentPosition()` se obtiene la posición actual del dispositivo (siempre que se tengan permisos). Como ellos mismos indican [23], Google no la recomienda ya que es más lenta y menos precisa. Sin embargo, es más **fácil** de implementar, requiere **menos permisos** y **menos librerías** de terceros.

react-native-background-geolocation, mauron85 [24]: Es una librería hecha por un usuario de GitHub. No parece defectuosa a primera vista, pero si se observa más en profundidad, tiene **muchas dependencias** (entre ellas Google Play API) y una difícil implementación. Esto complica mucho el desarrollo. Además, cuando se discutía con el responsable, se llegó a la conclusión que **no necesitamos una gran precisión**, sino que lo importante del proyecto es investigar sobre el tema de las

notificaciones push. Este tipo de librerías quedan desactualizadas al poco tiempo por falta de manutención.

react-native-fused-location, MustansirZia [25]: Utiliza la librería de Google Fused, que según sus creadores proporciona la localización **más precisa** [26]. De nuevo, es una librería que requiere de **otras de terceros**, más implementacion y más permisos.

React Native Firebase [27]: Esta librería se acabó usando para la conexión con Firebase. Se eligió esta ya que habían **pocas librerías** que ofrecieran una fácil conexión con el servicio de Google, y además ésta es **la única** que permite acceder al servicio de Push Notifications.

En cuanto a obtención de geolocalización, se acabó utilizando la propia de React Native (la misma en React para la versión Web). Se decidió usar esta ya que, como comentaba, lo que le interesaba al equipo de investigación SANS **es más la técnica de notificaciones push** que la precisión de la geolocalización. Se eligió la primera opción por simplicidad, documentación y **versatilidad** con la versión web.

5. Descripción técnica

Tras la investigación de alternativas, se decidió la siguiente estructura. En la **Figura 2** se puede ver un esquema de la arquitectura.

Se ha creado un servidor remoto para controlar el servidor de Firebase. Éste envía peticiones a un endpoint para que el servicio de Google realice el envío de notificaciones a un token.

El servidor Firebase envía estas notificaciones a los dispositivos. Los dispositivos, si tienen la función activada, contestan con un paquete de datos que con la geolocalización del móvil.

El servidor Firebase recibe estos paquetes de datos y los guarda separados por “experimentos” en una base de datos. Cada experimento tiene un código identificador único.

5.1. Librería

Es el producto final del proyecto. Consiste en una carpeta con un par de archivos: un worker para el background y la librería en sí. Incluye también la documentación. Puede funcionar sin demasiadas dependencias, adaptándose a la aplicación deseada.

5.1.1. Interfaz

Se detallan a continuación las funciones disponibles para llamar:

`startService()`

Enciende el servicio. Esto activará la recepción de notificaciones push. **Devuelve una Promise[13].**

`stopService`()

Detiene el servicio y desactiva la recepción de notificaciones.

`getUsername`()

Devuelve el nombre de usuario que se envía en el paquete. **Devuelve un string.**

`getToken`()

Devuelve el token que se envía en el paquete. Éste es un token propio, no el de Firebase. **Devuelve un string.**

`setUsername(newUsername)`

Establece el nombre de usuario para enviar a la BD.

`setToken(newToken)`

Establece el token propio para enviar a la BD. De nuevo, no es el token de Firebase, sino uno propio.

`registerToken()`

Registra el token **de Firebase** en la aplicación. **Devuelve una Promise<string>.**

[13] "Cuando se llama a una función asíncrona (*async*), ésta devuelve un elemento Promise. Cuando la función async devuelve un valor, Promise se resolverá con el valor devuelto. Si la función async genera una excepción o algún valor, Promise se rechazará con el valor generado." [28]

5.1.2. Funcionamiento interno

La clase se llama GeolocationService. Es ésta la que tiene todas las funciones disponibles para llamar. Como nota, en la versión web hay que pasar un objeto de configuración, así como la clave del servidor para conectarse a Firebase. Se explica más adelante.

Se detallan a continuación el funcionamiento de las funciones principales:

`constructor()`

Construye el objeto. Inicializa las conexión con con el servidor Firebase, establece las claves, y crea los objetos para referenciar a la base de datos.

También gestiona los parámetros de Token y Username (los propios). Es obligatorio que un usuario tenga un Username, pero no un token. El programador puede escribirlos con las funciones **setUsername()** y **setToken()**. Si no lo hace, el Username será un *string* aleatorio de 10 caracteres.

`registerToken()`

Obtiene el Token de **Firebase**. Éste será usado para referirse al dispositivo, y poderle enviar las notificaciones push.

`onRefreshToken()`

Cada vez que el token se regenera, llama a **registerToken()** de nuevo. Usa la función de Firebase para detectar cuando se regenera el token.

`startService()`

Gestiona la llegada de notificaciones push (llamados **mensajes**). Activa el *listener* de los mismos. Cada vez que llega un mensaje, comprueba si los parámetros de petición son correctos (tiene que llegar con el parámetro **getLoc = "true"**). Entonces, llama a **sendLocation(id)**.

`stopService()`

Detiene el *listener* de mensajes.

`sendLocation(id)`

El **id** es el identificador del experimento (se le envía desde el servidor, viene en el mensaje). Es opcional. Si se le pasa un ID, se guardará en la rama del experimento en la base de datos. En caso contrario, simplemente se almacenará la posición en la raíz de la BD.

Usa la llamada **navigator.geolocation.getCurrentPosition** de React/React Native discutida antes, para obtener la geolocalización. En caso que el sistema devuelva la posición sin problemas, envía a la base de datos un paquete con los siguientes datos: *timestamp, dateTime, latitude, longitude, coordsAccuracy, altitude, altitudeAccuracy, speed, heading, username, token, error* (false).

Si hay un error al obtener la geolocalización, el paquete será *timestamp, dateTime, error* (true)*, description*.

`getUsername()`

Devuelve un string en la versión Web y una **Promise<string>** en la App.

Lee el nombre de usuario del localStorage en caso de ser Web, o del AsyncStorage[14] en caso de ser la App. Si es la app, la función devuelve una **Promise** ya que será

`getToken()`

Devuelve un string en la versión Web y una **Promise<string>** en la App.

Lee el token de usuario del localStorage en caso de ser Web, o del AsyncStorage en caso de ser la App.

`setUsername(newUsername)`

En la App devuelve una **Promise<string>**.

Guarda el nuevo nombre de usuario en el localStorage en caso de ser Web, o en el AsynStorage en caso de ser la App.

`setToken(newToken)`

En la App devuelve una **Promise<string>**.

Guarda el nuevo token del usuario en el localStorage en caso de ser Web, o en el AsynStorage en caso de ser la App.

[14] https://facebook.github.io/react-native/docs/asyncstorage

5.2. Aplicación Android

Como he comentado, se creó también una aplicación de ejemplo en para React Native que funciona para Android. Ésta muestra un funcionamiento básico de las llamadas a la librería.

5.2.1. Instalación y setup

Para el desarrollo o testeo de esta app, hay que instalar el siguiente software:

- Instalar Node.JS[15]
- Instalar Watchman[16]
- Instalar React Native[17] en la pestaña "React Native CLI Quickstart"
- Instalar el Starter Kit[18] de RNfirebase, en caso que queramos desarrollar de cero.

Para que la aplicación funcione en segundo plano, añadir la siguiente línea en el archivo **AndroidManifest.xml**:

```
<service android:name="io.invertase.firebase.messaging.RNFirebaseBackgroundMessagingService" />
```

Para ejecutar la aplicación, moverse a la carpeta root desde un terminal. Abrir dos ventanas. En la primera, ejecutar: `$ npm start`

En la segunda, una vez cargada la primera, ejecutar: `$ npm run android`

5.2.2. Aplicación modelo

La aplicación muestra un botón para activar y desactivar el servicio. Primero creamos el botón, y el estado de la aplicación.

```
import React from 'react';
import {Button, Text, View} from 'react-native';
import GeolocationService from '../library/GeolocationService';

export default class Main extends React.Component {
   constructor(props) {
       super(props);

       this.state = {
          serviceOn: false,
          buttonColor: '#cddc39',
```

[15] NodeJS, core del lenguaje de programación: https://nodejs.org/en/

[16] Watchman, para actualizar la App en tiempo real mientras se desarrolla: https://facebook.github.io/watchman/

[17] React Native, el framework usado para la App: https://facebook.github.io/react-native/docs/getting-started

[18] RNFirebase, la librería comentada anteriormente para conectarse a Firebase: https://github.com/invertase/react-native-firebase-starter

```
            fcmToken: '',
        };

    }

    render() {
        return (
            <View>
                <Text style={{paddingBottom: 16}}>
 Press the button to enable the FCM service
                </Text>
                <Button
                    onPress={this.toggleService}
                    title={this.state.serviceOn ?
                          'Turn service off' : 'Turn service on'}
                    color={this.state.serviceOn ?
                           '#f44336' : '#cddc39'}
                 />
                <Text>
 {this.state.serviceOn ? 'Service is on' : 'Service is off'}
                </Text>
            </View>
        );  }}
```

Tabla 7. Código inicial de la app de ejemplo

Lo primero en ejecutarse será el constructor. Por ello, al final crearemos un objeto **controller**, que definiremos fuera del constructor.

```
controller;

constructor(props) {
   ...
   this.controller = new GeolocationService();
}
```

Tabla 8. Adición del controller al código

A continuación añadimos la función que se llamará cuando la app se cargue la vista. En ella registramos el token. Es de tipo **async** ya que el **regsterToken** es asíncrono.

```
async componentDidMount(): void {
   this.controller.registerToken().then(token => {
       this.setState({fcmToken: token});
       console.log('Registered!', token);
   });
}
```

Tabla 9. Función componentDidMount en React

Finalmente creamos la función **toggleService()** que será llamada cuando el usuario presione el botón. **setState()** cambiará el estado del botón, y el indicador global para saber si el servicio está activado o no. El **token** es inventado.

```
toggleService() {
   this.setState({
       buttonColor: this.state.serviceOn ? '#cddc39' : '#f44336',
       serviceOn: !this.state.serviceOn,
```

```
        });

        if (!this.state.serviceOn) {
            // Turn on
            this.controller.startService().then(() => {
                console.log('Turning on, waiting for messages');
            });
            this.controller.setToken('Leoijowefe3nx23x4-23x4234');
        } else {
            // Turn off
            this.controller.stopService();
            console.log('Turning off')
        }
    }
```

Tabla 10. Función toggleService()

Por último, hay que vincular esta nueva función que hemos hecho a la función. En el constructor añadimos una línea:

```
    this.toggleService = this.toggleService.bind(this);
```

Tabla 11. Bind del toggleService()

5.3. Aplicación Web

Como he comentado antes también, el proyecto incluía la petición de crear un código de ejemplo para la versión web. A pesar de ser bastante parecido al entorno de React Native, hay unos pequeños cambios a tener en cuenta.

5.3.1. Instalación y setup

Para el desarrollo o testeo de esta app, hay que instalar el siguiente software:

- Instalar Node.JS (el mismo que para la aplicación)
- Instalar Watchman (el mismo que para la aplicación
- Instalar ReactJS[19]

Para ejecutarla, moverse a la carpeta root desde un terminal y escribir:

```
$ npm start
```

[19] ReactJS es el framework usado para el desarrollo de la app web: https://reactjs.org/docs/getting-started.html

5.3.2. Aplicación modelo

La aplicación muestra un botón para activar y desactivar el servicio. Primero creamos el botón, y el estado de la aplicación.

```
import React from 'react';
import GeolocationService from './library/GeolocationService';

export default class Main extends React.Component {
   controller;

   constructor(props) {
       super(props);

       // Define the initial state.
       this.state = {
           serviceOn: false,
           buttonColor: '#cddc39',
           fcmToken: '',
       };
   }

   render() {
       return (
           <div>
               <p style={{paddingBottom: 16}}>
                  Press the button to enable the FCM service
               </p>
               <button
                   onClick={this.toggleService}
                   color={this.state.serviceOn ?
                          '#f44336' : '#cddc39'}>
                   {this.state.serviceOn ?
                    'Turn service off' : 'Turn service on'}
               </button>
               <p>Token: {this.state.fcmToken}</p>
               <p>
      {this.state.serviceOn ? 'Service is on' : 'Service is off'}
               </p>
           </div>
       );
   }
}
```

Tabla 12. Código inicial de la app de ejemplo

Lo primero en ejecutarse será el constructor. Por ello, inicializamos un objeto **controller**, que ya hemos definido fuera del constructor. En la versión web, hay que pasar dos parámetros al constructor: el archivo de configuración, y la clave del servidor. El objeto de configuración se puede conseguir directamente desde Firebase. Luego se explica cómo.

```
        // Paste here the Firebase Configuration.
        let firebaseConfig = {
            apiKey: "YOUR_API_KEY",
            authDomain: "DOMAIN.firebaseapp.com",
            databaseURL: "https://DOMAIN.firebaseio.com",
            projectId: "YOUR_ID",
            storageBucket: "DOMAIN.appspot.com",
            messagingSenderId: "YOUR_ID",
            appId: "YOUR_APP_ID"
        };

        // Paste here the public key
        let publicKey =
'YOUR_KEYGQDDY0TMOMigxxRWUkzu0ed1qyIqUq0gl3jX7Dhn5BOZgTBvuw4lncqlOxDCuGdKP10
ENN600BL0NJJJN_H_2I';

        // Define a new GeolocationService controller.
        this.controller = new GeolocationService(firebaseConfig, publicKey);
```

Tabla 13. Configuración de Firebase

A continuación añadimos la función que se llamará cuando la app se cargue la vista. En ella registramos el token. Es de tipo **async** ya que el **regsterToken** es asíncrono.

```
        async componentDidMount(): void {
            // In this moment we want to register the token and control the
returned value.
            this.controller.registerToken().then(token => {
                this.setState({fcmToken: token});
                console.log('Registered!', token);
            });}
```

Tabla 14. Función componentDidMount de React

Finalmente creamos la función **toggleService()** que será llamada cuando el usuario presione el botón. **setState()** cambiará el estado del botón, y el indicador global para saber si el servicio está activado o no. Ésta es exactamente igual que en la versión para React Native. El **token** es inventado.

```
        toggleService() {
            this.setState({
                buttonColor: this.state.serviceOn ? '#cddc39' : '#f44336',
                serviceOn: !this.state.serviceOn,
            });

            if (!this.state.serviceOn) {
                // Turn on
```

```
            this.controller.startService().then(() => {
                console.log('Turning on, waiting for messages');
            });
            this.controller.setToken('Leoijowefe3nx23x4-23x4234');
        } else {
            // Turn off
            this.controller.stopService();
            console.log('Turning off')
        }
    }
```

Tabla 15. Función toggleService

Por último, hay que vincular esta nueva función que hemos hecho a la función. En el constructor añadimos una línea:

```
    this.toggleService = this.toggleService.bind(this);
```

Tabla 16. Bind of the toggleService

5.4. Firebase

El servidor de Firebase es un gran e importante componente de la aplicación. Es el encargado de enviar las notificaciones y guardar los datos. Si bien tiene un uso muy sencillo, hay que hacer unos pocos pasos de configuración antes.

Primero navegaremos a la Consola de Firebase[20]. Creamos un nuevo proyecto, elegimos un nombre y los servicios que queremos.

Debemos ir a Configuración → Añadir aplicación. Ahí podemos copiar el objeto de configuración que usamos en la versión web (**punto 5.3.2**).

5.4.1. Push Notifications

Para activar las notificaciones push es necesario activar “Cloud Messaging”. Una vez activado, habrá que navegar de nuevo a la Configuración → Cloud Messaging → Generar Par de Llaves.

Esto generará la llave pública que necesitamos en la aplicación.

[20] Firebase Console: https://console.firebase.google.com/u/0/

5.4.1. Real Time Database

Hay que activar también Realtime Database. Para ello navegamos a la pestaña, y clicamos en activar. Las coordenadas se guardarán aquí. Si especificamos un ID, se guardarán en la rama **coords**. En caso contrario, se guardarán bajo **coordinates**.

5.5. Servidor

Finalmente llegamos al servidor. A través de éste, el técnico se comunicará con Firebase. El servidor es muy parametrizable, y permite tener mucho control sobre frecuencia y demás aspectos. Debido a que la comunicación es mediante un CLI (Command Line Interface), se ha elegido usar una librería [29] para hacerlo más agradable a la vista y fácil de usar. La configuración se puede poner en un fichero **config.json** en la raíz de la app.

5.5.1. Interfaz

El comando principal es `node app.js [opciones].` Las opciones pueden ser:

`--interval Xs | Xm | Xh`

Establece el intervalo de tiempo para enviar las notificaciones. Solo se puede usar una magnitud. Segundos es **s**, Minutos es **m** y Horas es **h**. La X es un número.

`--config path/to/config.json`

Path al fichero de configuración. Por defecto, busca en **./config.json**.

`--token myToken`

En caso de no especificarlo en el fichero **config.json**, el token se puede especificar con esta opción.

`--key myKey`

En caso de no especificarlo en el fichero **config.json**, la key se puede especificar con esta opción.

`--once`

En caso de querer enviar sólo una notificación, usar esta opción.

El fichero de configuración debe tener este formato:

```
{
    "key": "YOUR_KEY",
    "token": "YOUR_TOKEN"
}
```

Tabla 17. Fichero de configuración

5.5.2. Funcionamiento interno

El programa no es excesivamente complejo.

Al principio, hace una comprobación de todos los parámetros. En caso que algo no sea correcto, muestra un mensaje de error y explica cómo se usa el comando.

A continuación crea una variable mensaje con esta estructura:

```
var message = {
    to: token,
    data: {
        getLoc: "true",
        id: experimentID
    },
    priority: 'high'
}
```

Tabla 18. Mensaje enviado a Firebase

La variable **experimentID** es la fecha de hoy traducida a milisegundos. Se usa priority **high** para que todo funcione correctamente en **segundo plano**. La variable **token** es la que se pasa a través de parámetros, o lee del archivo.

Usando la librería comentada antes para los colores informamos al usuario de los parámetros pasados, y que se inicia el envío de mensajes.

Después de eso, con la función **setInterval** se va llamando a la función **sendNotification()** con la frecuencia indicada:

```
function sendNotification() {
    iteration++
    first = true

    fcm.send(message, (err, resp) => {
        if (first) {
            first = false
            if (err)
                console.log(chalk.bold.red('Error sending message '), err);
            else
                console.log(chalk.bold.green('Message ' + iteration) + ' - ' +
chalk.gray(new Date().toLocaleString()))
}})}
```

Tabla 19. Función sendNotification

6. Análisis de datos

6.1. Introducción

Se ha mencionado anteriormente que los datos de geolocalización se guardarán en la nube, en una base de datos de Google Firebase [3]. Queremos pues, antes de entregar la librería que nos ha sido encargada, asegurarnos de la **veracidad** y **fiabilidad** de los datos.

Si bien en la teoría, todo debería funcionar correctamente, en la práctica es más difícil. Nos enfrentamos a riesgos de **compatibilidad**, de **permisos**, o de la gestión que el sistema operativo hace de las **tareas en segundo plano**. Por eso, en este capítulo se diseñarán una serie de experimentos para comprobar que los datos que se están recolectando sean, efectivamente correctos.

Hay que tener en cuenta que estos experimentos se han hecho a lo largo del tiempo, empezando **desde los inicios del desarrollo de la librería**. Mediante estos primeros experimentos, no sólo fui capaz de comprobar si los datos eran correctos, pero también de entender aspectos clave del funcionamiento de las Notificaciones Push, que veremos más adelante.

6.2. Objetivos

Los principales objetivos de la investigación son los siguientes:

- **Configuración:** Queremos asegurarnos de cómo afectan las diferentes opciones de configuración para una aplicación sobre el rendimiento de ésta.
- **Precisión:** Queremos comprobar que las coordenadas que se están enviando sean, dentro de unos parámetros, correctas.
- **Segundo plano**: Queremos asegurarnos, también, que el sistema operativo no *mata* la aplicación después de un cierto tiempo.

Hay que tener en cuenta también las especificaciones iniciales de la librería. Lo más importante es que funcionase en segundo plano, y que pudiéramos obtener las localizaciones. Sin embargo, temas como el consumo extremo de batería, extrema precisión de las coordenadas, o rendimiento, **no eran prioritarias**.

6.3. Metodología

Se han diseñado los siguientes experimentos para cada uno de los objetivos indicados anteriormente

6.3.1. Experimento E1

Este ensayo nos indicará cuál es la **mejor configuración** de los parámetros del Sistema Operativo, y cuáles hacen que nuestra aplicación funcione mejor o peor. Éste se ha realizado en las primeras etapas del desarrollo de la librería.

El experimento consistirá en una tabla con **diferentes parámetros** que se irán modificando. Luego se enviarán **distintos tipos de mensajes** y se comprobará el resultado.

Se repetirá hasta que se encuentren los valores adecuados que buscamos.

6.3.2. Experimento E2

Esta prueba nos indicará si los datos del GPS **son acertados**.

Para probarlo, se creará un experimento en el cual, desde el servidor se enviarán notificaciones push en intervalos de **15 segundos**. El teléfono móvil se irá **moviendo caminando** por ciertas calles, y volverá al punto de partida.

Se repetirá dos veces para asegurar la fiabilidad.

Finalmente, para analizar los datos se hará un **mapeo sencillo de los puntos en un mapa,** y se comprobará si coincide con la realidad. El experimento durará unos 10 minutos, por lo que se espera tener aproximadamente $10 \times 4 = 40$ muestras.

Sabremos que funciona correctamente si los puntos tramados en el mapa coinciden con el camino seguido por el dispositivo móvil.

6.3.3. Experimento E3

Con este ensayo, queremos comprobar la **duración del segundo plano**, y asegurarnos que funciona correctamente.

Se hará un experimento en el que el servidor enviará notificación push con un intervalo de **1 minuto**. Se dejará aproximadamente unas **8h**, tiempo considerado suficiente para asegurarse que el proceso no muere.

Se repetirá dos veces para asegurar la fiabilidad.

Al final, se analizarán el número de notificaciones **recibidas** en la base de datos, el número de **errores** que llegan, y se **comparará** con el número de notificaciones enviada por el servidor. En este caso, no se mapearán los datos. El experimento durará unas 8h, por lo que se espera tener aproximadamente *8* x *60 = 480* muestras.

Sabremos que funciona correctamente si recibimos un número exactamente igual de mensajes en la base de datos, que los enviados por el servidor. Obviamente, contamos con que haya un margen de error, ya que por un motivo u otro, o bien no llegarán todos los mensajes a Firebase, o el teléfono inteligente no recibirá todas las notificaciones push. Por ello, consideraremos que un error del 20% es razonable.

6.4. Resultados

6.4.1. Experimento E1

Dispositivo: OnePlus 5T [37] / Android 8 - OxygenOS / Snapdragon™835.

Try	1	2	3	4	5	6	7
FCM Frequency	15''	15''	15''	15''	15''	15''	15''
Foreground	Yes	No	No	No	No	No	**Yes**
Screen	On	On	On	On	On	On	Off
Battery Optimization	No	Yes	No	No	**Yes**	Yes	Yes
Priority	-	-	-	-	-	**high**	high
App opened	Yes	Yes	Yes	No	Yes	Yes	**No**
Plugged	Yes	Yes	Yes	Yes	Yes	Yes	**No**
Moving	No	No	No	No	No	No	Si
First try working time	17'	1'	8'	73'	1'	65'	**240'**
Second try working time	17'	1'	8'	75'	1'	65'	239'
Average	17'	1'	8'	74'	1'	65'	240'
Comments	Dejó de funcionar al abrir otra app.			Detenido manual-mente.		Detenido manual-mente.	

Tabla 20. Resultados del experimento E1.

De éste experimento sacamos la conclusión de que lo que hizo que la aplicación funcionase correctamente en segundo plano, no era la configuración del teléfono, sino la configuración del servidor.

Nos dimos cuenta que lo que marcó la diferencia es añadir un parámetro en los mensajes FCM. Cómo se puede ver en la sección **5.5.2. Funcionamiento Interno** (Servidor), se debe añadir el campo *priority* y ponerle el valor *high* para que las notificaciones lleguen incluso cuando la app está funcionando en segundo plano.

6.4.2. Experimento E2

Una vez recolectados los datos, se ha creado una página HTML muy sencilla, basada en un código abierto de CodePen [38]. Con ésta, podemos ver un seguido de puntos en un mapa. Para ello sólo tenemos que cargar los datos en un objeto JSON, y luego añadir los puntos al mapa, mediante la API de Google Maps.

Para cargar los datos, no hace falta crear un nuevo programa. Usaremos las herramientas proporcionadas por los editores de texto modernos para selección de múltiples cursors [39].

El siguiente código JavaScript referencia un elemento en el HTML que tiene el ID *googlemaps*:

```
      var map;

      var json = [
       { "latitude" : "41.4107815", "longitude" : "2.1418102" },
       { "latitude" : "41.4108031", "longitude" : "2.1418238" },
         ...
       { "latitude" : "41.4021136", "longitude" : "2.1371985" },
      ];

      function initialize() {

       var mapOptions = {
         zoom: 6,
         center: new google.maps.LatLng(41.402806, 2.136535)
       };

       map = new google.maps.Map(document.getElementById('googlemaps'),
mapOptions);

       for(var i = 0; i < json.length; i++) {
         var marker = new google.maps.Marker({
           position: new google.maps.LatLng(json[i].latitude,
json[i].longitude),
           map: map,
         });
       }
      }

      google.maps.event.addDomListener(window, 'load', initialize);
```

Tabla 21. Código del programa para mapear latitudes y longitudes en un mapa.

Finalmente, podemos ver los puntos de las recolecciones de datos. En verde, el recorrido realizado en la realidad, y en rojo los puntos recolectados por la aplicación. La aplicación ha sido ejecutada en segundo plano y sin ninguna configuración especial en cuanto al consumo de batería.

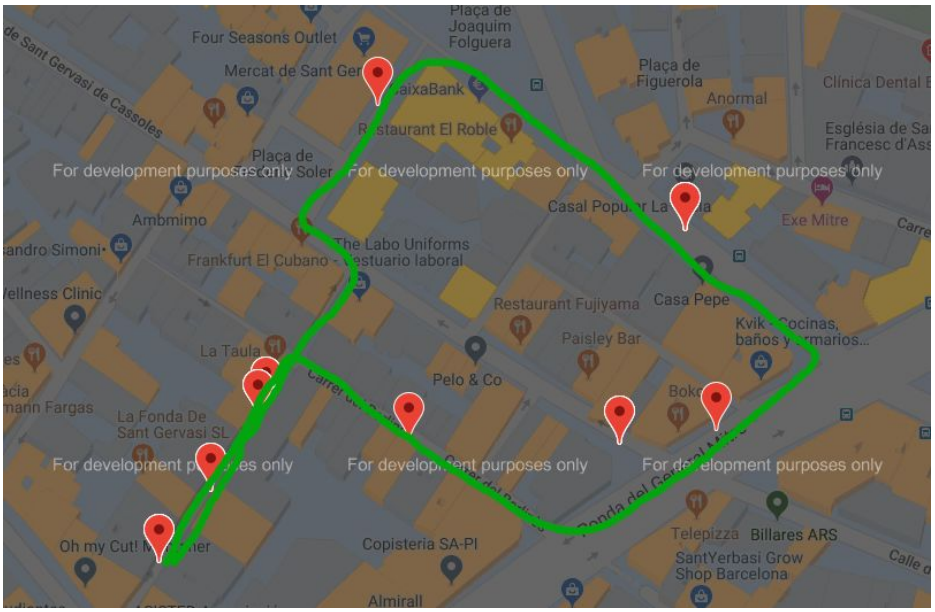

Figura 8. Experimento E2.1 - Mapa recorrido real en verde y puntos obtenidos en rojo

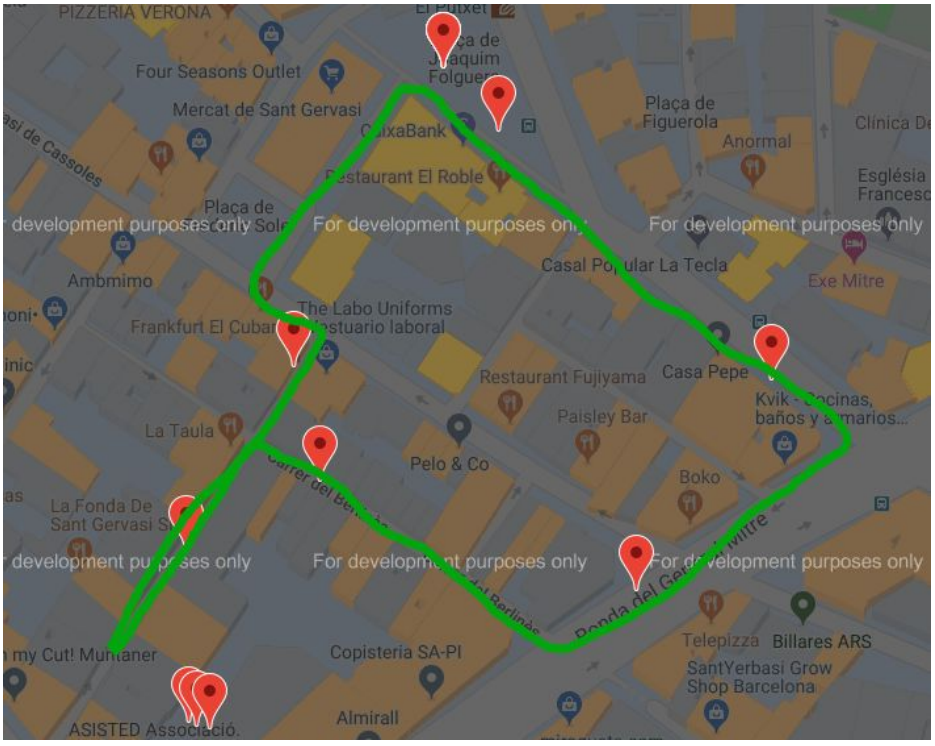

Figura 9. Experimento E2.2 - Mapa recorrido real en verde y puntos obtenidos en rojo

Cabe decir que si bien esperábamos 40 puntos, no se aprecian en la imagen. La explicación más probable para esto es que, como hemos visto anteriormente [42],

Android aproxime la ubicación, de forma que dos o más puntos tienen los mismos valores de *latitud* y *longitud*, y se vean superpuestos.

En conclusión podemos decir que estamos satisfechos con los resultados. Puesto que la librería se usará para detectar movimiento entre zonas de ciudades, el área cubierta en este experimento es relativamente pequeña. Es por eso que aunque falten algunos puntos debido a fallos de comunicación (véase experimento E3), el número de ubicaciones obtenidas es satisfactorio.

6.4.3. Experimento E3

Una vez tenemos los datos, analizaremos los resultados. En esta prueba nos interesaba conocer el funcionamiento en segundo plano. De nuevo, no es necesario crear un programa nuevo: con las herramientas de selección inteligente [39], obtenemos la siguiente tabla.

- Los **Envíos del servidor** son las notificaciones que ha enviado el programa *server* al servidor de Firebase.
- Los **Registros BD** es el número de registros que se han guardado en la base de datos,
- **Errores** es el número de mensajes que han llegado con el campo **error** a true, es decir, que no han conseguido obtener la localización.
- **Porcentaje errores** es el porcentaje de errores sobre **los envíos del servidor**, contando también los errores de transmisión entre *server* y Firebase.

Experimento	E3.1	E3.2
Duración	8h 19m	8h 31m
Envíos del servidor	497	504
Registros BD	490	499
Errores	86	79
Porcentaje errores	17,3%	15,6%

Tabla 22. Comparativa del experimento E3.1 y E3.2.

Vemos que hay un número de errores relativamente elevado, aunque no llega a significar el 20% que nos habíamos establecido como límite. De todas formas, se han encontrado respuesta a los diferentes errores:

Los errores de envío entre **servidor** y **base de datos** son debido a pérdidas de comunicación. En algunos casos, puesto que el programa **servidor** se está ejecutando desde un ordenador portátil con red **WiFi**, éste se ha desconectado de la red. De esta forma, como se puede ver en la imagen inferior, no se ha conseguido emitir el mensaje. Sin embargo, aunque el dispositivo móvil se encuentre sin cobertura, Firebase se asegurará de entregar el mensaje una vez restablezca la conexión.

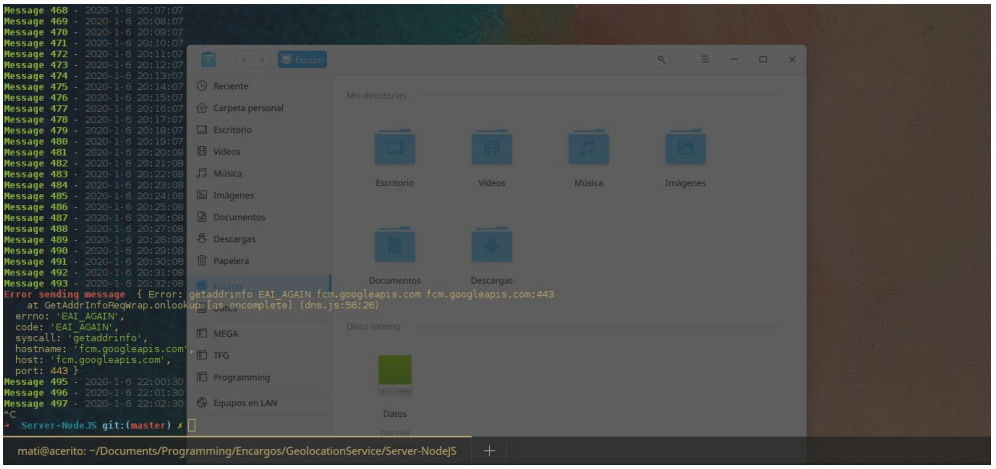

Figura 10. Error de envío debido a una desconexión de la red.

Los otros errores, en la base de datos se ven de la siguiente manera:

Figura 11. Error de obtención de geolocalización.

La razón es que en ese momento la señal GPS no estaba disponible. Por lo tanto, no es posible obtener la geolocalización.

En conclusión, podemos afirmar que en general funciona bien, y salvo algunos errores de comunicación, la librería **cumple con su función**. Cabe destacar, también, que el experimento se ha realizado con la app en segundo plano y sin modificaciones en la configuración de energía. Por lo tanto, podemos dar esta prueba por exitosa.

7. Informe de sostenibilidad

En todo proyecto de ingeniería es importante realizar una evaluación de sostenibilidad en los **tres aspectos principales**: dimensión económica, dimensión social, y dimensión ambiental. No se puede empezar un proyecto sin tener en cuenta las implicaciones que éste tendrá sobre la sociedad y el entorno en el cual se desarrolla.

Se incluye la matriz de sostenibilidad:

	PPP	Vida útil	Riesgos
Ambiental	9,15 kWh[21]	0,49 kWh/mes[22]	0 kWh
Económico	1237,14 €	Sueldo programadores	Sueldo programadores
Social	Ver punto 7.4	Impacto positivo en la comunidad científica (7.4)	Mala utilización del software

Tabla 22.1. Matriz de sostenibilidad.

[21] En el tiempo que dura el desarrollo del proyecto, dos meses.
[22] Por cada mes y usuario que se use la app.

7.1. Autoevaluación

Después de realizar la encuesta propuesta por la asignatura de GEP, se puede hacer un resumir en que en la FIB **hemos aprendido una gran cantidad sobre sostenibilidad económica, ambiental y social**, aunque como siempre, se puede mejorar. Me he dado cuenta de que conocemos muchos indicadores, y en general acostumbramos a pensar en la sostenibilidad ambiental y social cuando diseñamos un producto o servicio TI. Si bien se podría pensar más en la **economía circular** y en cómo darles vida a los productos una vez hayan terminado su ciclo de vida útil.

Por otra parte, al menos a nivel personal, acostumbro a pensar **más en el aspecto social que en el ambiental.** Es decir, solemos pensar mucho en el producto final al que va a ir nuestra aplicación, los usos que se le van a dar. Solemos pensar en si se le dará un uso bélico o para vulnerar los derechos de privacidad, pero pocas veces pensamos en la energía que puede gastar una cierta aplicación o producto TI, los recursos que se usan al desarrollar un producto o el tratado posterior a su vida útil.

7.2. Dimensión económica

Debido a que se trata de una librería con especificaciones muy concretas, la puesta en producción y su vida útil se valoran de la misma manera. Si se desease hacer un seguimiento con el tiempo, podríamos incluir los cálculos oportunos de actualización, pero no es el caso. De todas formas, sería interesante que, a nivel personal **se pueda ir adaptando la librería a la actualidad del software**, ya que como hemos visto, ahora mismo hay cosas que todavía no se pueden hacer, pero en un **futuro será posible.**

Más allá del desarrollo de la librería, dependerá de la empresa y/o programadores que la utilicen, los costes una vez puesta en marcha. En todo caso será importante cómo siga el desarrollo de los programadores para valorar su impacto económico una vez finalizado el alcance del proyecto actual.

7.3. Dimensión ambiental

A nivel ambiental, se puede reflejar que no es un proyecto que, ni para su puesta en producción, ni para su vida útil tenga un impacto ambiental excesivamente grande, ya que al final se trata de una aplicación.

Se puede **aproximar un gasto de lo que ha costado el desarrollo**, a nivel ambiental, haciendo un cálculo a nivel de los dos meses que dura el desarrollo del proyecto

- PC: 50W x 1,76h/día = 2,64 kWh/mes = 5,28 kWh en dos meses
- Android: 5W x 3,33 h/día = 0,49 kWh/mes = 0,99 kWh en dos meses
- iPhone: 5W x 3,33 h/día = 0,49 kWh/mes = 0,99 kWh en dos meses
- iMac: 100W x 0,33 h/día = 0,99 kWh/mes = 1,89 kWh en dos meses

Por lo tanto, el consumo energético final es de 9,15 kWh en dos meses.

7.4. Dimensión social

Es importante pensar qué implicaciones sociales tendrá este proyecto una vez acabado. Por eso, a nivel personal **le doy mucha importancia a saber exactamente para qué se utilizará la librería,** dado que no seré yo quien la explote. Los datos de geolocalización son material muy sensible, y saber dónde está una persona, o incluso un colectivo de personas, **puede ser crucial**.

A primera vista, dado que el proyecto de SANS busca cruzar los datos de grupos de personas con mapas tridimensionales de polución de la ciudad, tendrá un impacto **social positivo a largo plazo en la vida de estas personas**, una vez el equipo haya sacado sus conclusiones de la investigación. Sin embargo, si esta librería cayera en manos inadecuadas, alguien podría utilizarlo para fines diferentes a los que se pensó originalmente. Es por eso, que por ejemplo se ha decidido mantener el repositorio de GitHub privado. De todas maneras, si una empresa quiere desarrollar un software como éste, no le será demasiado difícil, teniendo en cuenta que tienen muchos más recursos.

A nivel personal, la realización del TFG me ha ayudado a entender **cómo crear de cero un sistema cliente-servidor,** a aprender a programar en React Native, a hacer un seguimiento semanal del proyecto… Me ha ayudado mucho también aprender cómo hacer una buena planificación, y reflexión de las implicaciones del proyecto. Estos conocimientos adquiridos son importantes a nivel personal ya que aportan muchas cualidades que empresas u otros individuos valorarán positivamente en un futuro cercano.

7.5. Marco legal

Al desarrollar un producto tecnológico, y especialmente de captación de material sensible como la posición geolocalizada de un individuo, es importante informarse sobre la normativa vigente. No solo eso, sino que en mi opinión es importante también defender la privacidad e intimidad de todos los usuarios de cualquier aplicación.

Se analiza, a continuación, las leyes alrededor de la geolocalización de personas.

7.5.1. Derecho a la intimidad

Se puede encontrar en la Constitución Española [30], en el artículo 18 una explicación sobre la defensa de la intimidad personal y familiar:

> *1. Se garantiza el derecho al honor, a la intimidad personal y familiar y a la propia imagen.*
>
> *2. El domicilio es inviolable. Ninguna entrada o registro podrá hacerse en él sin consentimiento del titular o resolución judicial, salvo en caso de flagrante delito.*
>
> *3. Se garantiza el secreto de las comunicaciones y, en especial, de las postales, telegráficas y telefónicas, salvo resolución judicial.*
>
> *4. La ley limitará el uso de la informática para garantizar el honor y la intimidad personal y familiar de los ciudadanos y el pleno ejercicio de sus derechos.*

Artículo 18 de la Constitución Española [31]

El punto más interesante son el tercero y el cuarto. En ellos se describe el **secreto** de las comunicaciones, así como **la limitación** de la informática para garantizar la intimidad personal. ¿Significa esto que es ilegal recabar cualquier dato íntimo de una persona? Según **artículo 197 del Código Penal,** es ilegal cualquier cesión de datos de carácter identificativo sin el consentimiento de la persona [32].

> *El que, para descubrir los secretos o vulnerar la intimidad de otro, sin su consentimiento, se apodere de sus papeles, cartas, mensajes de correo electrónico o cualesquiera otros documentos o efectos personales, intercepte sus telecomunicaciones o utilice artificios técnicos de escucha, transmisión, grabación o reproducción del sonido o de la imagen, o de cualquier otra señal de comunicación, será castigado con las penas de prisión de uno a cuatro años y multa de doce a veinticuatro meses.*

Punto 1 del artículo 197 del Código Penal [32].

Por lo tanto, sin ser un experto en derecho, se llega a la conclusión que resultaría ilegal y penado, así como una vulneración de los derechos la recabación de datos de geolocalización **no anónimos**, de una persona.

7.5.2. K-anonimidad

Ahora bien, ¿Qué es la anonimidad? Según la Real Academia Española, se se define como "*Indiferenciado, que no destaca de la generalidad*". Por lo tanto, significa que no se puede diferenciar o identificar.

Sin embargo, en el 2002 Latanya Sweeney trabajó en un estudio sobre el anonimato real en los datos: "*Se dice que un conjunto de datos publicados tiene la propiedad de k-anonimato (o es k-anónimo) si la información de todas y cada una de las personas contenidas en ese conjunto es idéntica al menos con otras k-1 personas que también aparecen en dicho conjunto.*" [32]

Esto se puede ver en un ejemplo. Supongamos la siguiente lista de pacientes de un hospital en India [33].

Nombre	Edad	Género	Estado	Religión	Enfermedad
Ramsha	30	Mujer	Tamil Nadu	Hindú	Cáncer
Yadu	24	Mujer	Kerala	Hindú	Infección
Salima	28	Mujer	Tamil Nadu	Musulmán	TB
Sunny	27	Hombre	Karnataka	Parsi	Sano
Joan	24	Mujer	Kerala	Cristiano	Cardiológico
Bahuskana	23	Hombre	Karnataka	Budista	TB
Rambha	19	Hombre	Kerala	Hindú	Cáncer
Kishor	29	Hombre	Karnataka	Hindú	Cardiológico
Johnson	17	Hombre	Kerala	Cristiano	Cardiológico
John	19	Hombre	Kerala	Cristiano	Cardiológico

Tabla 23. Ejemplo de datos sin anonimizar

A esta tabla se le aplican dos transformaciones. La **eliminación** o **supresión**, marcada con un asterisco "*", consiste en eliminar columnas de la tabla para hacer a los individuos anónimos. La **generalización** consiste en reemplazar un dato por una categoría, como por ejemplo "*menor de 20 años*". En la siguiente tabla, no se puede identificar a ninguno de los pacientes. Estos datos cumplen con el **2-anonimato**

Nombre	Edad	Género	Estado	Religión	Enfermedad
*	20 < Edad ≤ 30	Mujer	Tamil Nadu	*	Cáncer
*	20 < Edad ≤ 30	Mujer	Kerala	*	Infección
*	20 < Edad ≤ 30	Mujer	Tamil Nadu	*	TB
*	20 < Edad ≤ 30	Hombre	Karnataka	*	Sano
*	20 < Edad ≤ 30	Mujer	Kerala	*	Cardiológico
*	20 < Edad ≤ 30	Hombre	Karnataka	*	TB
*	Edad ≤ 20	Hombre	Kerala	*	Cáncer
*	20 < Edad ≤ 30	Hombre	Karnataka	*	Cardiológico
*	Edad ≤ 20	Hombre	Kerala	*	Cardiológico
*	Edad ≤ 20	Hombre	Kerala	*	Cardiológico

Tabla 24. Tabla de datos 4-anonimizado

Este tipo de anonimato nos podría servir para un estudio como el que el grupo SANS ha indicado que quiere realizar. Sin embargo, aún existen algunas vulnerabilidades en este tipo de anonimato.

Pongamos que por un motivo (por ejemplo, por los datos catastrales o porqué lo conocemos), sabemos que John, de 19 años vive en Kerala. Sabemos también que él está en esta base de datos, porqué va a este hospital. Entonces, podemos **estar seguros** que John tiene un problema cardiológico.

7.5.3. GDPR

En abril de 2016 se aprobó en el Parlamento Europeo el **Reglamento General de Protección de Datos** (GDPR por sus siglas en inglés) [37]. En esta ley se describe claramente cómo debe ser el tratado de datos de cualquier **institución o persona** de la Unión Europea. A raíz de ésta, se pueden extraer 7 puntos [36] para ser conforme y cumplir con estas nuevas regulaciones:

1. **Obtener consentimiento:** El usuario tiene claro qué datos estamos obteniendo.
2. **Notificación de brechas:** En caso de una brecha de información, la organización deberá notificar al usuario en menos de 72 horas.

3. **Derecho al acceso de los datos:** El usuario tiene el derecho de solicitar sus datos de forma electrónica y gratuita.
4. **Derecho al olvido:** El usuario tiene derecho a pedir la eliminación completa de sus datos.
5. **Portabilidad de datos:** El usuario es el dueño de sus datos. Pueden usar el punto 3 para pedir los datos, pero además los pueden reutilizar fuera de la organización.
6. **Diseño de privacidad:** Los sistemas de recolección de datos deberán ser diseñados *orientados a la privacidad* desde el principio.
7. **Encargado de protección de datos:** Dependiendo del tamaño de la empresa, se necesitará un DPO (Data Protection Officer).

7.5.4. Conclusión

Después de haber revisado ciertas leyes y conceptos que pueden afectar al uso de esta librería, y de nuevo admitiendo que no soy un experto en leyes, se llega a la conclusión que la librería está adaptada a las leyes.

El razonamiento es el siguiente. Al ser una librería "de terceros", no soy yo quien lo implementa. Tampoco soy yo quien guarda ni trata los datos. Por lo tanto, podemos decir que la librería **estaría cumpliendo con todas las leyes**.

Sin embargo, es importante remarcar que es el programador **que use nuestra librería**, quién deberá hacerse cargo de la conformidad con la legislatura. Se sugieren las siguientes medidas:

1. **Informar al usuario** de la recolección de sus datos.
2. **Anonimizar** los datos recogidos (a través del username o token).
3. **No relacionar** nunca los datos guardados con personas físicas.
4. Permitir al usuario final **la eliminación** completa de todos sus registros.
5. Comprobar la **seguridad** de la base de datos, sus contraseñas y configuraciones.
6. En caso de una brecha de seguridad, **informar al usuario** final.
7. Dar la posibilidad a los usuarios de **descargar sus datos** en cualquier momento.

8. Conclusión

8.1. Técnico y académico

Una vez acabado el proyecto, y mirándolo con perspectiva, estoy orgulloso de decir que **ha sido exitoso** y todo el duro trabajo **ha valido la pena**. El proyecto ha **cumplido los objetivos** propuestos, y funciona como es de esperar. Además, la memoria adjunta a la realización del proyecto le aporta el valor añadido de la investigación.

Como he comentado en contadas ocasiones durante el informe, la geolocalización en segundo plano para aplicaciones en React Native es aún **un tema muy nuevo** y que poca gente ha trabajado sobre ello. Por lo tanto, se puede afirmar que este proyecto **aporta un valor importante al mundo tecnológico, académico**, y a cualquier actor implicado.

Se han puesto en práctica, además las **competencias esperadas**, en una medida bastante ajustada a las especificaciones del TFG: se ha entendido cómo funciona el grupo de investigación SANS; se ha **diseñado** y desplegado una plataforma de software dentro de unos **parámetros de calidad** establecidos; se ha **dirigido y planificado** la gestión del proyecto; se ha demostrado, en cierta medida, la **seguridad y privacidad** del proyecto.

Al final, se ha **concebido un sistema** y aplicación basado en la integración de **diferentes tecnologías**, que es un aspecto importante de mi especialidad, TI. Éste ha incluido, en cierta medida, la integración de **diferentes comunicaciones** también.

Durante el proyecto también se han puesto en práctica numerosas competencias **adquiridas durante la carrera**, y que por supuesto no hubiera sido posible adquirir por otras vías. Tampoco hubiera sido posible realizar un proyecto de calidad sin estas competencias. Y no sólo técnicas, como la correcta **programación**, experimentación, implementación de sistemas, **diseño de interfaces** o almacenamiento y análisis de los datos; también se han puesto en práctica muchas **competencias transversales**, como la iniciativa, curiosidad, pensamiento crítico, capacidad de hablar en público, o la expresión verbal y escrita.

Por último, debido a que el proyecto se usará para la investigación de grupos de riesgo, aportará una mejor calidad de vida a las **personas vulnerables**.

8.1.3 Justificación de los objetivos y la relación con el grado

Se adjunta a continuación una lista de los objetivos y competencias más trabajados durante la realización del trabajo:

- CTI1.1: Se ha demostrado comprender, en cierta medida, **cómo funciona una organización** de investigación; la relación con un cliente, entender sus necesidades y demandas en el ámbito tecnológico. Desde la definición de las características del proyecto, a tener un producto acabado.
- CTI1.4: Con profundidad, se ha ha **diseñado, desplegado, integrado, construido y gestionado un proyecto tecnológico** dentro de unos parámetros adecuados. Además, se ha evaluado y mantenido dentro de los requisitos económicos pactados.
- CTI2.1: De la misma forma, se ha **dirigido, planificado y coordinado** la gestión del software a implementar en bastante profundidad.
- CTI2.3: Como ya se ha demostrado en el apartado de privacidad, se ha conseguido **garantizar la seguridad y privacidad de los usuarios**, así como dar herramientas a los desarrolladores para asegurarlo.
- CTI3.1: Al final del proyecto, se ha podido ver como se ha **concebido un sistema y aplicación basado en tecnologías de la información**.
- CTI3.3: Se ha conseguido **implementar y configurar diferentes servicios** para un bien y objetivo común.
- CTI3.4: Se ha **diseñado** un software con complejos **componentes de comunicaciones**.

Como se ha comentado antes, se han puesto en práctica muchos conocimientos adquiridos durante el Grado en Ingeniería Informática (GEI) en la Facultad de Informática de Barcelona (FIB - UPC).

- Correcta **programación e implementación** de una librería en el entorno de un proyecto (asignaturas de PRO1, PRO2, EDA, PROP).
- Utilización y gestión de **comunicaciones**, **APIs** y sistemas de **geolocalización** (PI, XC, IM, SO, SOA).
- **Almacenaje** y tratado de la **información** (BD, SDX)
- **Diseño de interfaces de usuario** (IDI).
- Entendimiento de las configuraciones y **componentes hardware** que han intervenido en el proyecto (APC).
- **Securización** y garantización de la privacidad (SI).
- **Gestión** de un proyecto informático (GEP, VPE, PTI).

8.2. Personal

A nivel personal, este trabajo ha sido un proceso importante en mi aprendizaje. Por una parte, me ha enseñado a **organizarme, a plantear un proyecto de cero,** a ser más exigente conmigo mismo. He aprendido a escribir un documento de más de 60 páginas en un ámbito académico y formal, a preservar los estilos y la concordancia del informe, y a ponerme unas ciertas **fechas con unos objetivos a alcanzar**.

He aprendido a **planificar detalladamente** lo que voy a hacer, cosa con la que no me había encontrado seriamente hasta este momento. Sin embargo, la realización del TFG me ha aportado también la habilidad de saber cambiar y saber maniobrar a tiempo para **no perder recursos** con aspectos que no lo justifican.

Por otro lado, he aprendido a **hacer un presupuesto**, una factura, a declarar el IVA y **gestionar toda la burocracia** que ello conlleva. Además, me ha abierto la mente a un campo que nunca había considerado trabajar: **la investigación**. No estoy seguro si será mi futuro trabajo, pero después de esta etapa, **definitivamente no la descarto**.

En el aspecto más profesional, sin embargo, no puedo decir que haya aprendido a trabajar en equipo. De todas formas, en la carrera hemos hecho tantos proyectos en grupo que no considero que fuera una necesidad esencial. Sí me ha aportado, sin embargo, la experiencia de trabajar en una relación con alguien que tiene unos **requerimientos y unas exigencias**, así como plazos y otras demandas.

Además, el proyecto realizado destacará durante mucho tiempo en mi CV. Al fin y al cabo, **estoy muy satisfecho** con el trabajo realizado. Creo que he sabido llevarlo a cabo de **forma exitosa.** También estoy orgulloso de la parte que más me atemorizaba: la **redacción de la memoria**. Sin embargo, en mi humilde opinión pienso que en un grado razonablemente bueno, he **sabido defender y justificar** todas las decisiones llevadas a cabo durante la concepción de la librería.

8.3. Futuro y mejoras

Por supuesto, a pesar de estar satisfecho con el proyecto, siempre se puede mejorar y pensar en progresos para el futuro. Si se le dedicase más tiempo al trabajo, estos son los aspectos o características que me gustaría incluir:

- **Fuera de conexión:** Si el dispositivo móvil se desconecta, Firebase se encarga de entregar la Push Notification. Sin embargo, sería interesante que una vez conectado, el paquete que se envía a la base de datos contuviera información sobre el tiempo de desconexión del teléfono, para que en su posterior tratado se pueda tener en cuenta.
- **Modo autónomo:** Si se quisiera añadir más soporte a la desconexión de la red, se podría incluir un modo autónomo, en el cual el servidor envíe

una notificación push con el tiempo de intervalo al móvil. Entonces, la librería podría encargarse de registrar la geolocalización, aún no teniendo conexión con internet. Una vez recuperada la conexión, podría enviarse todos los datos recopilados.

- **Alternativas de geolocalización:** Como se ha comentado, ahora se usa en la librería un sistema de obtención de geolocalización basado en GPS. Sin embargo, se podría optimizar la obtención de ubicaciones basadas en CellID [45] (obtención de ubicación de la antena a la que está conectado el dispositivo), mediante triangulación WiFi [46] (obtención de SSID conocidos en lugares determinados), u otros métodos vistos en el grado en ingeniería informática.

9. Bibliografía

[1] SANS, «Statistical Analysis of Networks and Systems (SANS) Research Group,» Available: http://sans.ac.upc.edu.

[2] A. S. N. Service, «Amazon Simple Notification Service,» 2019. Available: https://aws.amazon.com/es/sns/. [Último acceso: Mayo 2019].

[3] F. C. Messaging, «Firebase Cloud Messaging,» 2019. Available: https://firebase.google.com/docs/cloud-messaging/?hl=es-419. [Último acceso: Mayo 2019].

[4] A. Olarte, «Medium,» Mayo 2019. Available: https://medium.com/wolox-driving-innovation/react-native-integrando-push-notifications-con-firebase-cloud-messaging-9476ad5cf08. [Último acceso: Mayo 2019].

[5] A. Amin, «Medium,» Julio 2018. Available: https://medium.com/@anum.amin/react-native-integrating-push-notifications-using-fcm-349fff071591. [Último acceso: Mayo 2019].

[6] «React Native Firebase,» Available: https://rnfirebase.io.

[7] R. F. C. Messages, «React Native Firebase,» Available: https://rnfirebase.io/docs/v4.3.x/messaging/receiving-messages#4)-(Optional)(Android-only)-Listen-for-FCM-messages-in-the-background . [Último acceso: Mayo 2019].

[8] E. G. t. workers, «GitHub,» Nov 2018. Available: https://github.com/w3c/ServiceWorker/issues/745. [Último acceso: Mayo 2019].

[9] M. Rouse, «TechTarget,» Septiembre 2019. Available: https://searchsoftwarequality.techtarget.com/definition/hybrid-application-hybrid-app. [Último acceso: Septiembre 2019].

[10] M. Rouse, «TechTarget,» Julio 2018. Available: https://searchmobilecomputing.techtarget.com/definition/push-notification. [Último acceso: Septiembre 2019].

[11] ReactJS, «ReactJS,» Available: https://reactjs.org.

[12] R. Native, «ReactNative,» Available: https://facebook.github.io/react-native/.

[13] Postman, «Postman,» Available: https://www.getpostman.com. [Último acceso: Abril 2019].

[14] M. Foundation, «Usando Web Workers,» Available: https://developer.mozilla.org/es/docs/Web/Guide/Performance/Usando_web_workers. [Último acceso: Abril 2019].

[15] MacServiceBCN, Available: https://macservicebcn.com/alquiler-ordenadores-mac/. [Último acceso: Noviembre 2019].

[16] PadInTheCity, Available: https://www.padinthecity.com/es/alquiler-iphone. [Último acceso: Noviembre 2019].

[17] Wikipedia, «Desarrollo ágil de software», Available: https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software. [Último acceso: Noviembre 2019].

[18] Ionic «Cross-Platform Mobile App Development», Available https://ionicframework.com/ [Último acceso: Noviembre 2019].

[19] Angular, «Superheroic JavaScript MVW Framework» Available https://angular.io/ [Último acceso: Noviembre 2019].

[20] Apache Cordova, Available https://cordova.apache.org/ [Último acceso: Noviembre 2019].

[21] VueJS «The progressive JavaScript Framework», Available https://vuejs.org/ [Último acceso: Noviembre 2019].

[22] M. Rajput «Top Mobile App Development Framework in 2019», Available https://www.mindinventory.com/blog/mobile-app-development-framework-2019/ [Último acceso: Noviembre 2019].

[23] React Native «Geolocation», Available https://facebook.github.io/react-native/docs/geolocation [Último acceso: Noviembre 2019].

[24] mauron85 «react-native-background-geolocation» (GitHub Repo), Available https://github.com/mauron85/react-native-background-geolocation [Último acceso: Noviembre 2019].

[25] MustansirZia «react-native-fused-location» (GitHub Repo), Available https://github.com/MustansirZia/react-native-fused-location [Último acceso: Noviembre 2019].

[26] Google, «FusedLocationProviderApi», Available https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderApi [Último acceso: Noviembre 2019].

[27] React Native Firebase, Available https://rnfirebase.io/ [Último acceso: Noviembre 2019].

[28] Mozilla Web Docs «Función Async», Available https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias/funcion_asincrona [Último acceso: Noviembre 2019].

[29] Chalk (Repo), Available https://www.npmjs.com/package/chalk [Último acceso: Noviembre 2019].

[30] J. D. Meseguer González «Derechos fundamentales afectados por la geolocalización», Available https://elderecho.com/derechos-fundamentales-afectados-por-la-geolocalizacion [Último acceso: Noviembre 2019].

[31] Constitución Española, Available https://www.derechoshumanos.net/constitucion/index.htm [Último acceso: Noviembre 2019].

[32] Artículo 197 del Código Penal, Available https://www.conceptosjuridicos.com/codigo-penal-articulo-197 [Último acceso: Noviembre 2019].

[33] Wikipedia ES «k-anonimato», Available https://es.wikipedia.org/wiki/K-anonimato [Último acceso: Noviembre 2019].

[34] Wikipedia EN «k-anonymity», Available https://en.wikipedia.org/wiki/K-anonymity [Último acceso: Noviembre 2019].

[35] S. Saltis «GDPR Explained In 5 Minutes: Everything You Need to Know», Available https://www.coredna.com/blogs/general-data-protection-regulation [Último acceso: Noviembre 2019].

[36] GDPR Available: https://gdpr.eu [Último acceso: Noviembre 2019].

[37] OnePlus 5T Available: https://www.oneplus.com/es/support/spec/oneplus-5t [Último acceso: Enero 2020].

[38] Simen «Google Maps marker from JSON» Available: https://codepen.io/schikulski/pen/urozH [Último acceso: Enero 2020].

[39] Sublime Text «Multiple Selection with the Keyboard» Available: https://www.sublimetext.com/docs/3/multiple_selection_with_the_keyboard.html [Último acceso: Enero 2020]

[40] Uber Available: www.uber.com [Último acceso: Enero 2020]

[41] Google «Google Maps Timeline» Available: https://support.google.com/maps/answer/6258979?co=GENIE.Platform%3DiOS [Último acceso: Enero 2020]

[42] Android Developers «Cómo conocer la ubicación más reciente» Available: https://developer.android.com/training/location/retrieve-current [Último acceso: Enero 2020]

[43] Wikipedia «Interfaz de programación de aplicaciones» Available: https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones [Último acceso: Enero 2020]

[44] React Native Documentation «Geolocation» Available: http://facebook.github.io/react-native/docs/geolocation.html [Último acceso: Enero 2020]

[45] Combain «Cell ID for positioning» Available: https://combain.com/about/about-positioning/cell-id-positioning/ [Último acceso: Enero 2020]

[46] red Zahradnik «An Explanation of Wi-Fi Triangulation» Available: https://www.lifewire.com/wifi-positioning-system-1683343 [Último acceso: Enero 2020]

10. Anexos

10.1. Gantt

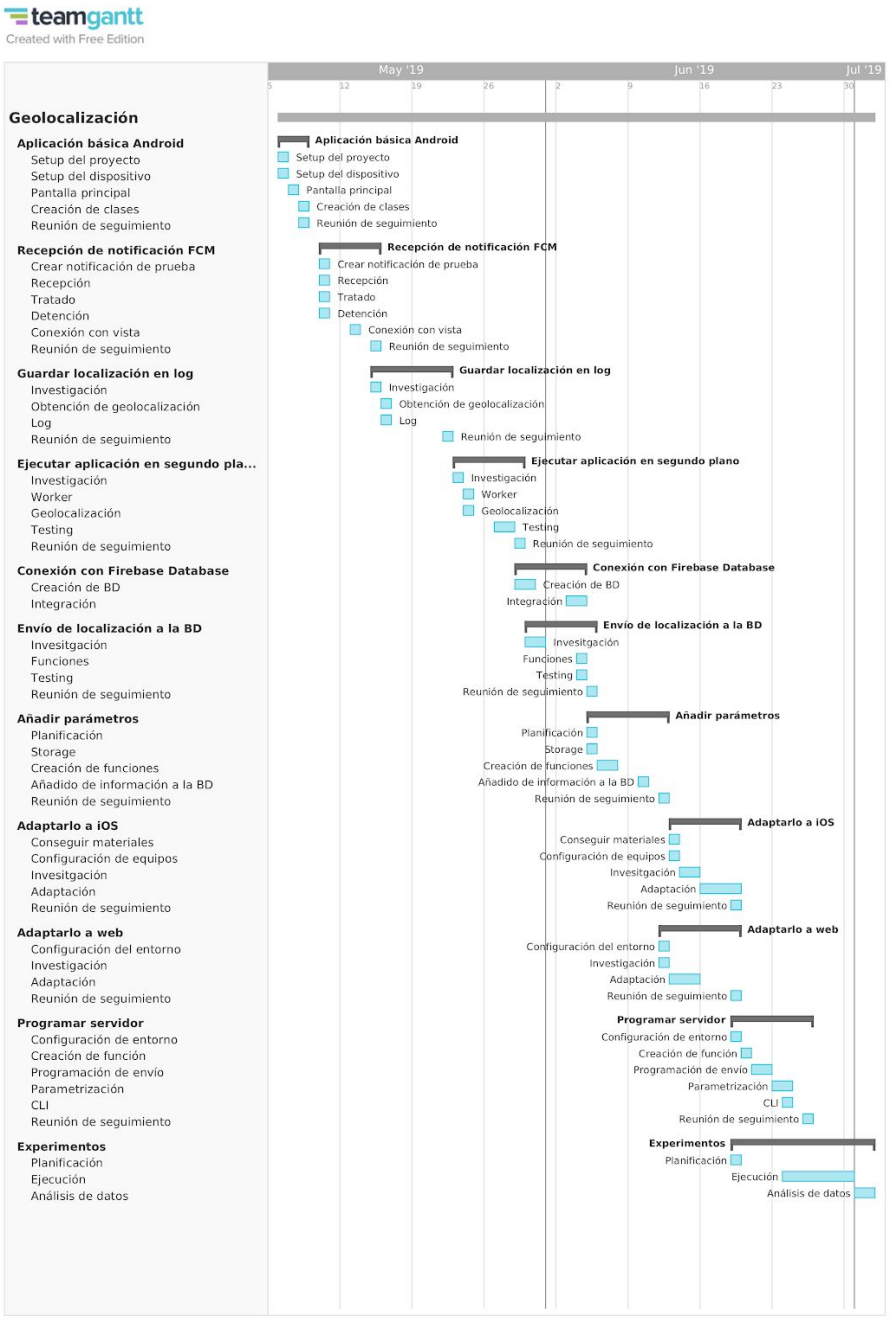

Figura 12. Diagrama de Gantt

10.2. Tareas

Fase	Stage	Tarea	Tiempo (h)	Recursos						
				PC	Android	Mac	iPhone	FCM	Tutor	Matías
MVP (Producto Mínimo Viable)	Aplicación básica Android	T1 – Setup del proyecto	1	■	■					■
		T1.1 – Setup del dispositivo	0.5	■	■					■
		T2 – Pantalla principal	1	■	■					■
		T2.1 – Creación de clases	2	■	■					■
		T3 – Reunión de seguimiento	0.5						■	■
	Recepción de notificación FCM	T4 – Crear notificación de prueba	2	■	■			■		■
		T4.1 – Recepción	2	■	■			■		■
		T4.2 – Tratado	2	■	■			■		■
		T4.3 – Detención	1	■	■			■		■
		T5 – Conexión con vista	0.5	■	■			■		■
		T6 – Reunión de seguimiento	0.5						■	■
	Guardad localización en log	T7 – Investigación	2	■						■
		T8 – Obtención de geolocalización	3	■	■			■		■
		T8.1 – Log	1	■	■			■		■
		T9 – Reunión de seguimiento	1						■	■
	Ejecutar aplicación en segundo plano	T10 – Investigación	2	■						■
		T11 – Worker	2	■	■			■		■
		T11.1 – Geolocalización	1	■	■			■		■
		T12 – Testing	1.5	■	■			■		■
		T13 – Reunión de seguimiento	0.5						■	■
Detalles	Conexión con Firebase Database	T14 – Creación de BD	3	■				■		■
		T15 – Integración	3h	■	■			■		■
	Envío de localización a la DB	T16 – Investigación	2	■				■		■
		T17 – Funciones	4	■	■					■
		T18 – Testing	2	■	■					■
		T19 – Reunión de seguimiento	1						■	■
	Añadir parámetros	T20 – Planificación	2	■					■	■
		T21 – Storage	4	■	■					■
		T22 – Creación de funciones	4	■	■					■
		T23 – Añadido de información a la Bl	4	■	■			■		■
		T24 – Reunión de seguimiento	1						■	■
	Adaptarlo a iOS	T25 – Conseguir materiales	2			■	■			■
		T26 – Configuración de equipos	4			■	■			■
		T27 – Investigación	2	■		■				■
		T28 – Adaptación	11			■	■			■
		T29 – Reunión de seguimiento	0.5						■	■
	Adaptarlo a web	T30 – Configuración de entorno	3.5	■						■
		T31 – Investigación	1	■						■
		T32 – Adaptación	10	■						■
		T33 – Reunión de seguimiento	0.5						■	■
	Programar un servidor para el envío de mensajes	T34 – Configuración de entorno	2	■						■
		T35 – Creación de función	2	■						■
		T35.1 – Programación de envío	4	■	■					■
		T36 – Parametrización	4	■						■
		T37 – CLI	3.5	■						■
		T38 – Reunión de seguimiento							■	■
TFG	Experimentos	T39 – Planificación	2	■						■
		T40 – Ejecución	168	■	■					■
		T41 – Análisis de datos	5							■

Figura 13. Tabla de tareas

Índice de figuras

Índice de tablas